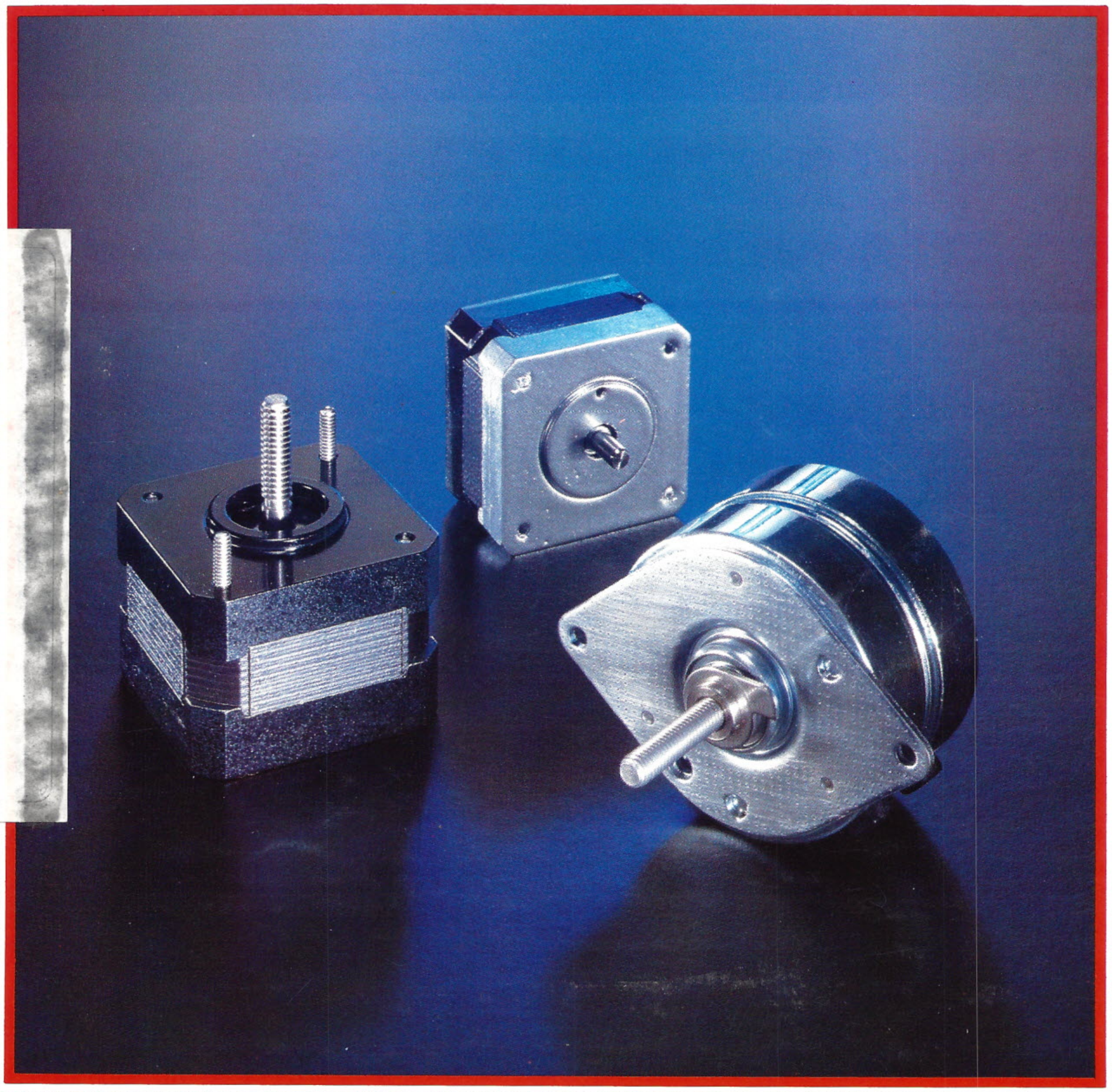
Intelligent Motor Control

DECEMBER 1984 VOL. 6 NO. 12 $3.00 USA/$3.50 CANADA

THE JOURNAL OF INTELLIGENT MACHINES

ROBOTICS AGE™



AMAROBOT™

Personal Robot Store

THE FUTURE IS AMAROBOT

SINK YOUR TEETH INTO SOMETHING SPECIAL

PRESENTING

THE AMAROBOT TURTLE

ONLY $249.00

Amarobot welcomes you to the infancy of Personal Robotics. Watch as we develop the industry, in much the same way as the personal computer field was developed 20 years ago. If you missed the beginnings then, or if computers still seem unfriendly to you now, here's a chance to sink your teeth into something special. Afterall, a robot is a computer, a much more friendly one — like a pet from another dimension of time and space. And if you're one of the captains on spaceship Earth who likes to set the trends, the Amarobot turtle is choice. Amarobot has designed an inexpensive turtle, one that grows with you as personal robotics grows. And you needn't ask, "What can it do?" Your Amarobot turtle can do almost anything. The only limitation is imagination, and imagination is unlimited — as unlimited as the future...

CURRENT AVAILABILITY: COMMODORE-64 & ATARI 800

BASIC TURTLE: software included, Programmable, Dual motor, Independent, Variable speed 2 1/0, Cable, Paddle drive-teach 4 prog. LEDs, Wheel encoder, Pulse width modulated

OPTIONS NOW AVAILABLE:

Vacuum attachment:	$35.00
Forth language:	$65.00
2 sets bumper switches - pair:	$19.95
32 outputs attachment:	$49.95

OPTIONS SOON AVAILABLE: Arm, Body, Voice, Vision, Logo language, Graphics, Infrared data link, RS232, Real time clock, Course manuals, Kit forms, Hall encoding, Biofeedback link (mind control)

Name ____________________

Address ____________________

City ____________ State ________ Zip ________

I've enclosed ________ / Charge my VISA __ M/C __

Card No. ____________ Expiration ________

Signature ____________________

AMAROBOT
1780 Shattuck Avenue
Berkeley, CA 94709
(415) 548-3554

VISA · MasterCard

DEALER INQUIRIES INVITED
CATALOGUE $2.00 REFUNDABLE

Publisher/Editorial Director
Carl T. Helmers, Jr.

Editor
Raymond GA Cote

Consulting Editor
Russ Adams

Directory Editor
Heidi Copeland

Production/Design
Nancy Estle

Production Assistant
Tobee Phipps

Circulation Manager
James E. Bingham

Dealer Accounts
Meredith Makela

Circulation Assistant
Trish Harper-Lentricchia

Typography
Maryellen Kelly

Receptionist
Vi Depres

Advertising Manager
Donna Louzier

Advertising Coordinator
Cheryl Wilder

Advertising Sales

Jerry Merrifield
Robotics Age Inc.
174 Concord Street
Peterborough, NH 03458
603-924-7136

Member of
Business Publications Audit
(SMA Division)

BPA

ROBOTICS AGE—(ISSN 0197-1905) is published monthly by Robotics Age Inc., Strand Building, 174 Concord Street, Peterborough, NH 03458, phone (603) 924-7136. Address subscriptions, changes of address, USPC Form 3579, and fulfillment questions to *Robotics Age* Subscriptions, 174 Concord Street, Peterborough, NH 03458. Second class postage paid at Peterborough, NH and at additional mailing offices. **POSTMASTER:** Send address changes to ROBOTICS AGE, 174 Concord St., Peterborough, NH 03458.

Subscriptions are $24 for one year (12 issues), $45 for two years (24 issues), $63 for three years (36 issues) in the USA and its possessions. In Canada and Mexico, subscriptions are $28 for one year, $53 for two years, $75 for three years. For other countries, subscriptions are $32 for one year, surface delivery. Air delivery to selected areas at additional charges, rates upon request. Single copy price is $3 in the U.S., $3.50 in Canada and Mexico, $4 in Europe, and $4.50 elsewhere. Foreign subscriptions and single copy sales should be remitted in United States funds drawn on a U.S. bank.

Address all editorial correspondence to the Editor at *Robotics Age*, Strand Building, 174 Concord Street, Peterborough, NH 03458. Opinions expressed by the authors of articles are not necessarily those of *Robotics Age*. To aid in preparation of acceptable articles, an Authors' Guide is available upon request, if accompanied by a self-addressed 8½ by 11 inch envelope with first class postage for 3 ounces. Unacceptable manuscripts will be returned if accompanied by a self-addressed envelope with sufficient first class postage. Not responsible for lost manuscripts or photos.

THE JOURNAL OF INTELLIGENT MACHINES

ROBOTICS AGE™

DECEMBER 1984 VOL. 6 NO. 12

EDITORIAL

2 Marketing Questions — by Raymond GA Cote

FEATURES

8 Motion Control for Automation and Robotics — by Jacob Tal and Wayne Baron
A mathematical description of digital control systems.

12 A Lisp-Based Robot Control System — by Doug Snead and John Roach
Part II: Location and Object References
Adding the ability to reference locations and names instead of precise vector definitions.

20 Robotix — by Mark J. Robillard
Part I: Examining the Pieces
A new, inexpensive construction kit for robot experimenters.

24 Introduction to Prolog — by Dean A. Schlobohm
Part II: A Simple Expert System
A high-level language for logic programming.

26 A Third-Generation Stepper Motor Controller — by Ed Klingman and Karen Moty
Part I: Intelligent Control
Single-chip microcontroller eases motor control design.

DEPARTMENTS

4 Calendar
6 Sensory Perceptions
19 Advertiser Index
33 New Products
36 Classified Advertising

About the cover: This cover, courtesy Berger Lahr of Jaffrey, New Hampshire, shows three two-phase stepping motors used as head-positioning actuators in high-capacity applications. From left to right, the first motor is used in a high-density tape backup drive. The second is for a high-capacity 3.5 inch format Winchester, and the third is utilized in an advanced technology tape drive still being developed. The technology and design techniques used to control high-density memory components is also useful in the world of high-precision, light industrial robots.

Editorial

BY RAY COTE

MARKETING QUESTIONS

I have recently received several telephone calls from graduate students taking surveys for courses covering business planning, market analysis, and even engineering evaluations. I'm always happy to answer their questions because I can discuss the answers they have already received (it's a wonderful, informal survey) and because I want to know what their questions are.

To date, most of the surveys asking about personal or home robots seem to make one fundamental error. They *assume* that the personal robot is right around the corner. Some surveys even assume it has already arrived.

The basic question which these surveys want to answer is, "How can we make money in the personal robot marketplace?" I am still waiting for the survey that asks, "Is there a personal robot marketplace? If so, how large? If not, how soon?" Although I firmly believe that personal robots will someday be a household item, I don't see it as a reality for the next five to ten years. We have yet to go through the agonizing, yet inevitable, development stage. After all, in this day and age you can't be a serious industry unless you have a "shake-out."

The real question facing potential personal robot manufacturers is this. Should a 1985 *personal* robot (not experimental, nor educational) try to take out the garbage or should it simply be a nifty, whizz-bang gizmo that is amusing to have about the house? The Heath Co. Consumer Division is hoping that the 1985 personal robot should simply be entertaining. Perhaps they are right. After all, the personal computer that was supposed to revolutionize the household simply brought entrancing (to some), colorful, shoot-em-down or blow-em-up video games to your television. (By the way, why isn't it a "personal" television set?)

Entertainment may well be the personal robot's initial role. A cute little box that runs around the house and says "Hello" to family and friends and shouts "Halt there" to intruders. Something to impress the neighbors who don't yet have one, and something to talk to when you just need to hear the sound of your own voice.

Although it is not the labor-saving, floor-scrubbing, garbage-toting machine envisioned as the modern solution to domestic help, the personal robot *has* succeeded in getting one foot (wheel?) in the door.

December 1984

3–4 December. Expert Systems: A Practical Application of Artificial Intelligence (Course #1120DC). George Washington University, Washington, DC 20052. Contact: J. Perkins, George Washington University, Washington, DC 20052, telephone (202) 676-8510.

A continuing education course designed to enhance and expand the technical skills of practicing professionals in the field of artificial intelligence. This course identifies the many areas in which expert systems are being applied and presents the fundamental concepts and components of systems. It presents the various design issues and discusses methods for testing, validation, and evaluation of systems. Implementation, management issues, and future trends are also examined.

4–7 December. International Conference on Robotics and Factories of the Future. Charlotte, NC. Contact: Mary Pat Young, University of North Carolina at Charlotte, UNCC Station, Charlotte, NC 28223, telephone (704) 597-2307.

The conference will provide guidelines for future planning, research, education, and factory design for higher productivity through robotics and CAD/CAM. The conference will have international participation from speakers and exhibitors.

11 December. Artificial Intelligence and Expert Systems. Sheraton Lincoln Inn, Framingham, MA. Contact: Lee Burgess, Rensselaer Polytechnic Institute, Troy, NY 12181, telephone (518) 266-6589.

This Executive Briefing session is designed by senior management for senior management personnel. It will provide the basic information needed to make decisions about when and where to use AI/ES in products and processes.

11–14 December. Various ASME Courses. Marriott Hotel, New Orleans, LA. Contact: ASME Professional Development Department, American Society of Mechanical Engineers, 345 East 47th St., New York, NY 10017, telephone (212) 705-7123.

Five short courses are offered by the ASME: Introduction to Computer Graphics, Computer-Aided Design and Manufacturing Technology, Management Skills Levels I & II, Project Management for Engineers, and Reducing Product Liability Exposure.

February 1985

11 February. Artificial Intelligence and Expert Systems. Marriott/Copley Boston, MA. Contact: Lee Burgess, Rensselaer Polytechnic Institute, Troy, NY 12181, telephone (518) 266-6589.

This Executive Briefing session is designed by senior management for senior management personnel. It will provide the basic information needed to make decisions about when and where to use AI/ES in products and processes.

26-28 February. Agri-Mation 1. Palmer House Hotel, Chicago, IL. Contact: SME Public Relations, One SME Drive, PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500.

This three-day exposition will feature demonstrations of a wide range of automated equipment used in the production of food, fiber, and biomass materials. Featured will be new developments in irrigation and drainage, field equipment, material handling, storage and distribution, harvesting, processing, and the latest computerized systems and services available to the agricultural manager, researcher, and engineer. Among the agricultural automation products on display will be data acquisition systems, measurement instruments, mobile robots, motors, programmable controllers, robotic components and systems, sensors, controls, transducers, and actuators.

March 1985

4–6 March. FutureCare. San Antonio, TX. The Gunter Hotel, San Antonio, TX. Contact: The University of Texas, Health Science Center at San Antonio, Occupational Therapy Program, 7703 Floyd Curl Drive, San Antonio, TX, 78284, telephone (512) 691-7555.

FutureCare is a three-day conference concerning the role of technology in rehabilitation. The sym-

Circle 11

Calendar

posium allows clinicians using the new technologies in rehabilitation to share their ideas, experiences, and research with others and also provides a forum for the exploration and discussion of a broad range of technological applications to clinical rehabilitation. Topics include cognitive rehabilitation, use of robot systems, patient evaluation and treatment, and computer-assisted instruction.

21–22 March. Computers and Young Children. University of Delaware, Newark, DE. Contact: Dr. Richard B. Fischer, Division of Continuing Education, University of Delaware, Newark, DE 19716, telephone (302) 451-8838.

This is the second national conference sponsored by the University of Delaware. The conference explores the opportunities and problems educators face as computers are introduced into educational programs. The emphasis is on programs for children 4–8 years old.

26–28 March. Vision '85. Cobo Hall, Detroit, MI. Contact: Jeff Burnstein, Robotic Industries Association, PO Box 1366, Dearborn, MI 48121, telephone (313) 271-7800.

Vision '85 will be the first major exhibition of machine vision systems and related equipment ever held. A comprehensive technical conference will accompany the show. RIA's sponsorship of Vision '85 is an outgrowth of the trade association's new emphasis on machine vision.

April 1985

22–25 April. LASERBOTICS: Combining Laser and Robot Technologies. Ann Arbor, MI. Contact: Steve Palma, SME Special Programs Department, One SME Drive, PO Box 930, Dearborn, MI 48121, telephone (313) 271-1500.

The program will feature speakers from the U.S., Europe, and Asia and will examine the latest advancements in combining lasers and robots to improve productivity. Co-chairperson Jack Lane, director of the Robotics Center at the GMI Engineering and Management Institute, and David Belforte, President of Belforte Associates, are preparing an agenda covering the latest developments in laser tooling, robotic part presentation, laser-guided robotics, fiber optics, laser/robot welding and inspection.

23–24 April. 1985 Conference on Intelligent Systems and Machines. Oakland University, Rochester, MI. Contact: Professor Nan K. Loh, Conference Chairman, Center for Robotics and Advanced Automation, School of Engineering and Computer Science, Oakland University, Rochester, MI 48063.

Technical papers are requested which reflect both advances and applications in all aspects of intelligent systems and machines. Suggested topics include: intelligent robots, machine intelligence, adaptive control and estimation, visual perception, artificial intelligence for engineering design, intelligent simulation tools, computer-integrated manufacturing systems, knowledge representation, expert systems, game theory and military strategy, interpretation of multisensor information, and automatic programming. Authors are requested to submit a 300- to 500-word abstract by 1 December 1984.

Sensory Perceptions

ROBOT REPORT. How fast will the personal robot industry grow? One hundred percent per year between 1983 and 1990, at which point it will constitute a $2.2 billion industry. This figure is taken from *The Robot Report*, a comprehensive, five-volume analysis of key marketing and technical issues in the personal robot industry conducted by Future Computing, Inc. According to David Wilson, Senior Analyst for Future Computing and author of the five-volume report, the personal robot industry today is much like the personal computer industry was ten years ago. It's in its infancy, and most people building personal robots are experimenters who don't really know the full range of capabilities and applications or the market possibilities for their inventions.

According to Wilson, the *Robot Report* takes a hands-on approach to engineering and design problems and addresses the marketing and distribution issues that companies will have to face in order to take advantage of the opportunities in the personal robot industry.

Volume 1 is devoted to marketing and distribution strategies. It analyzes 187 firms currently manufacturing personal robots, giving statistics such as size, sales trends, marketing organization, and engineering capabilities of each company. Volumes 2 through 4 discuss technological issues. Volume 2 details major robot subsystems and functions, including hands, arms, vision, voice synthesis, speech recognition, base units, and power supplies. Volume 3 emphasizes design and engineering problems and solutions pertinent to technological developments. Volume 4 analyzes a personal robot built by Future Computing in 1984. It explains general specifications and design criteria. The last volume of the *Robot Report* is an annotated bibliography covering significant books and papers applicable to the design of a personal robot.

The *Robot Report* is priced at $6,000 and is available from Future Computing, a unit of the McGraw-Hill Information Services Co., 8111 LBJ Freeway, Dallas, TX 75251.

RB REORGANIZES. RB Robot Corp. has filed its plan of reorganization, including provisions to satisfy creditors. President Joseph Bosworth announced the company's intention to merge with Actronix Corp of Dallas, Texas. The merger plan will be presented to RB Robot shareholders for approval. This decision negated a June 25 announcement that RB Robot had reached an agreement with Robotics International Corp. of Jackson, Michigan to form a limited partnership for the development and marketing of an obstacle-avoidance sensor system for personal robots. Under the reorganization, however, Robotics International will be granted minority ownership in the RB Robot/Actronix merged entity.

Actronix, a privately owned company incorporated in March 1983, has developed prototypes of two personal robots—the Actron Bear, an upright device with a 300-pound lifting capacity, and the Actron Wolf, a low-profile mobile security device.

HIGH-TECH CLOTHING. TechStyle, Inc. (the name is much more significant once you say it out loud) is marketing a machine designed to automatically sew pockets on trousers. If this doesn't sound like a major technological breakthrough, consider that this is the first time pockets have been sewn without skilled human intervention. The machine picks up facings that are the parts of the pocket structure, assembles them, registers them, and then sews them into proper position with *sewbots*. The company is also working on building a low-cost vision system to be used with a dedicated computer as part of a multipurpose apparel work station.

The ultimate goal is to create a totally automated apparel machine that could sew anything from men's suits to tents. The hope is that this type of machine

Sensory Perceptions

will help the American garment industry compete with foreign manufacturers.

SEATTLE ROBOTICS SOCIETY. The Seattle Robotics Society meets the last Thursday of each month. Its most recent newsletter describes a variety of member's machines. From the descriptions, the members seem to be actively creating their own robot systems. For more information, contact: Seattle Robotics Society, c/o United Products, Inc., 1123 Valley St., Seattle, WA 98109.

AI IC DESIGN. Gould, Inc. has been awarded a contract by the U.S. Army Electronic Devices Technology Laboratory, ERADCOM, for the development of an expert system as part of the U.S. Army's ongoing program to apply modern integrated-circuit technology to new systems and established Army equipment. A primary project task is to develop design tools and methods to aid in the redesign of electronic systems now becoming obsolete.

To do this, the specifications of the obsolete system must be captured in technology-independent terms. This requires software methods to translate the system specifications into current IC technologies. This method allows new technology to be introduced into existing systems in a more efficient manner and eliminates the need to use obsolete technology or completely discard systems.

REMCON UPDATE. In September, *Robotics Age* published a review of the Remcon Teach Robot. At that time, distribution was to start in mid-September. As of this writing (mid-October), Remcon has decided to redesign some of the electronics boards. This redesign has delayed the Teach Robot introduction into the United States.

Some people have called us to say that they have had difficulties reaching Remcon Electronics at its Nashville number, (615) 361-3936. This is the number of a small, local operation, which is still active, although someone is not always there to answer calls. For more information, send a written request for product literature to Remcon Electronics, Ltd., PO Box 148258, Nashville, TN 37214.

PATENT INFORMATION. Keeping up to date with pertinent information often means wading through reams of irrelevant papers. Optosonic Press has created a series of U.S. Patent reports that eliminate the trouble and expense of searching through the patent database. Their Patent Technology Portfolios cover topics from liquid crystal display devices to bar code technology and applications and magnetic bubble domain devices.

Particularly interesting is the portfolio titled *Industrial and Personal Robotics*. This portfolio contains 193 patents issued between 1977 and 1982. Selected topics include manipulators, universal joints, wrist construction, driverless vehicles, vision, speech synthesis, and control systems. More information about the Patent Technology Portfolios can be obtained from Optosonic Press, PO Box 883, Ansonia Station, New York, NY 10023.

MOTION CONTROL FOR AUTOMATION AND ROBOTICS

Jacob Tal
Wayne Baron
Galil Motion Control
1916-C Old Middlefield Way
Mountain View, CA 94043

An essential part of any automation or robotics system is the motion control element. This is the part that performs the actual motions in accordance with the desired commands.

There are two common motion control methods: open-loop control by step motors and closed-loop control by DC motors. Step motor control does not require any position-sensing devices, and therefore, results in lower system cost. However, step motors have several disadvantages which limit their performance.

The main disadvantage of step motors is their resonances. Step motors have a low-frequency resonance mode which results in an underdamped response. Step motors also have some mid-frequency resonances which may cause loss of synchronization at certain speeds. These disadvantages are compounded by the use of open-loop systems that do not observe the motor position and therefore cannot determine if synchronization has been lost. These disadvantages severely limit step motor motion control system performance.

An alternative approach is closed-loop control of DC and brushless motors. Such a control system utilizes a position sensor for feedback and uses the feedback to close the control loop around the motor. Closed-loop control systems require a controller that observes the motor position, closes the loop, and maintains loop stability. These functions can best be performed by a microprocessor.

In order to describe the state of the art of motion control systems, we start with a review of the design approaches.

DESIGN APPROACHES

The early motion control systems were based on analog circuitry. The elements of a basic analog control system are shown in Figure 1. In order to stabilize those systems, it was necessary to include velocity feedback, a continuous compensation network, or both. The availability of microprocessors enabled designers to use the microprocessor to close the position loop digitally.

Recently, a growing number of systems have incorporated digital control into their design; for example, the microprocessor is used to monitor the motor position and is also used to calculate the compensation values indicated by the feedback. Such a system is shown in Figure 2. Since microprocessor-based systems are very stable, there is no need for velocity feedback nor for an analog compensation system.

SYSTEM OPERATION

A digital motion control system can be represented by the block diagram shown in Figure 3. The host computer first determines where its motor is to be moved. It then sends the instructions to the micro-

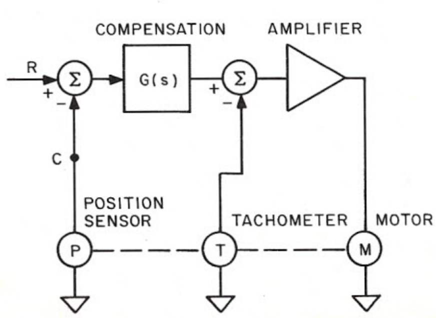


Figure 1. Diagram for analog position control system.

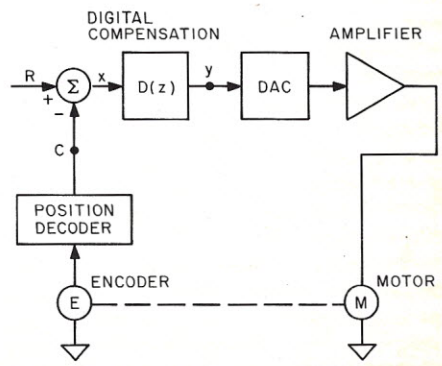


Figure 2. Diagram for digital position control system.

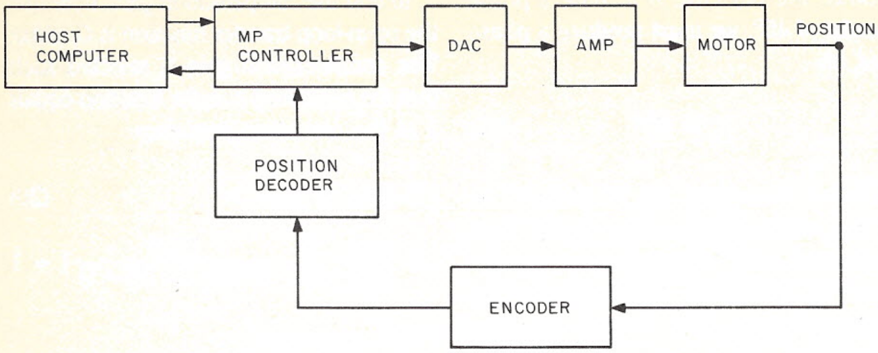


Figure 3. Digital motion control system elements.

processor controller. The controller also monitors the position feedback to determine the position error. The microprocessor controller determines the instruction to be sent to the motor based on the actual instruction from the host computer and the position error that has been detected. The microprocessor controller then transmits the appropriate command to the motor. This command is first sent to the D/A converter where it is converted from the digital form to an analog form. The analog command is then applied to the amplifier that drives the motor.

The motor position is fed back to the microprocessor controller by a position sensor. The most common position sensor is an incremental encoder that generates N encoder pulses per motor revolution. In order to sense the direction of motion, an incremental encoder is often used with two channels in quadrature. The output encoder signals are shown in Figure 4. Since every encoder cycle can be further divided into four intervals, the effective resolution of the encoder is $4N$ quadrature counts per revolution.

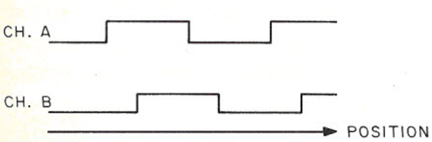


Figure 4. Output signals from incremental encoder.

The output from this incremental encoder is then applied to the position decoder which counts the encoder pulses and acts as a buffer between the encoder and the microprocessor.

SYSTEM MODELING

Figure 5 represents a functional block diagram of the mathematical model for a digital motion control system. The motor transfer function is represented by $M(s)$. This is the transfer function between the input voltage and the output position. A commonly used model is represented by:

$$M(s) = \frac{\theta}{V} = \frac{1/K_t}{s(sT_m+1)} \left[\frac{\text{rad}}{\text{volt}}\right] \quad (1)$$

Where K_t is the torque constant and T_m is the motor's mechanical time constant which is given by:

$$T_m = \frac{rJ}{K_t^2} \text{ [sec]} \quad (2)$$

r is the motor armature resistance and J is the combined moment of inertia for the motor and load. The amplifier is modeled as a gain, K_a, and the D/A converter is represented by a gain, K_d. The effective gain of an n-bit D/A converter with an output voltage of $\pm V_s$ is:

$$K_d = \frac{2V_s}{2^n} \left[\frac{\text{volt}}{\text{count}}\right] \quad (3)$$

The encoder is also represented as a constant gain, K_p. Since the encoder generates $4N$ counts per revolution, its effective gain is:

$$K_p = \frac{4N}{2\pi} \left[\frac{\text{count}}{\text{rad}}\right] \quad (4)$$

The desired position generator calculates a sequence of positions, R, that correspond to a desired profile. The controller compares the desired position, R, with the position feedback, C, to form the position error.

$$X = R - C \quad (5)$$

The position error signal is processed by a digital filter, $D(z)$, to form the motor command, Y. This is performed periodically with a sampling period, T.

$$D(z) = \frac{Y(z)}{X(z)} \quad (6)$$

A common form of digital filter is a lead compensation. Such a filter has the transfer function:

$$D(z) = C\,\frac{z-A}{z-B} \quad (7)$$

This digital filter is equivalent to a continuous filter with the transfer function:

$$G(s) = K\,\frac{s+a}{s+b} \quad (8)$$

The equivalence between the two transfer functions is:

$$A = e^{-aT} \quad (9a)$$

$$B = e^{-bT} \quad (9b)$$

$$C\,\frac{1-A}{1-B} = K\,\frac{a}{b} \quad (9c)$$

These mathematical definitions allow us to describe the system by a continuous model. This model is the basis for the following system analysis and design.

SYSTEM DESIGN

Consider a position control system with the parameters given in Table 1. The design objectives may be expressed in terms of two parameters: the crossover frequency, ω_c,

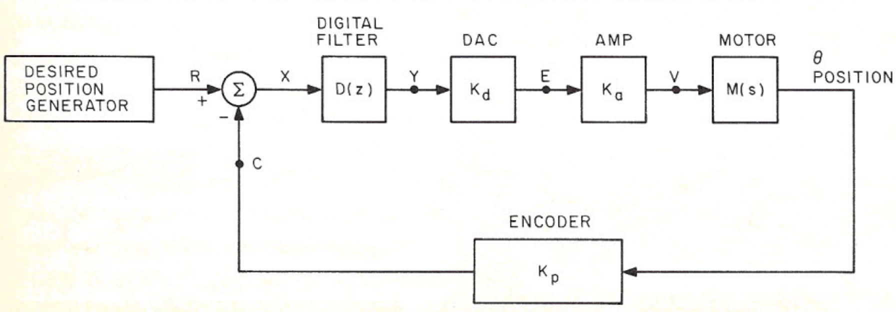


Figure 5. Mathematical model of the digital motion control system.

and the phase margin, θ_m. The crossover frequency is the frequency at which the open-loop gain equals one. This frequency is approximately equal to the final system bandwidth.

The phase margin θ_m determines the degree of stability and is commonly selected between 30° and 45° to produce a stable response without excessive overshoot.

For the design example given in Table 1, we require a crossover frequency of:

$$\omega_c = 125 \text{ rad/s} \tag{10}$$

and a phase margin of:

$$\theta_m = 45° \tag{11}$$

We start the design with the system model shown in Figure 6. All of the continuous elements of the system are combined into one function, *H(s)*:

$$H(s) = \frac{1740}{s(0.2S+1)} \tag{12}$$

The phase shift of *H(s)* at the crossover frequency is:

$$\phi_1 = \arg[H(j\omega_c)] \tag{13}$$

Here:

$$\phi_1 = \arg\left[\frac{1740}{j125(j0.2\times 125+1)}\right] = -90° - \tan^{-1}(25) = -178° \tag{14}$$

Since the objective is to have a phase margin of 45°, we must produce a phase lead of:

$$\phi_2 = 43° \tag{15}$$

Such a phase lead can be achieved by a lead compensation. The design procedure for such a compensation can be found in the books *Motion Control by Microprocessors* and *Control Systems Design.*

In order to achieve the required phase lead of 43°, the pole-zero ratio (assume *a* and *b* respectively denote the frequencies of the pole and the zero) needs to be approximately:

$$\gamma = \frac{b}{a} \cong 6.25 \tag{16}$$

For the most effective placement of the pole and zero, select *a* and *b* such that:

$$\omega_c = \sqrt{ab} \tag{17}$$

Since the crossover frequency is $\omega_c = 125$, the resulting pole and zero are:

$$\begin{aligned} a &= 50 \\ b &= 312 \end{aligned} \tag{18}$$

and the compensation is:

$$G(s) = K\,\frac{s+50}{s+312} \tag{19}$$

To find the compensation gain, note that the open-loop transfer function is *G(s)H(s)*. The compensation gain is selected such that the open-loop transfer function equals 1 at the crossover frequency.

$$|G(j\omega_c)\,H(j\omega_c)| = \tag{20}$$

$$\left| K\,\frac{j125+50}{j125+312}\;\frac{1740}{j125(j0.2\times 125+1)} \right| = 1$$

This results in:

$$K = 4.5 \tag{21}$$

and therefore:

$$G(s) = 4.5\,\frac{s+50}{s+312} \tag{22}$$

The function, *G(s),* is actually implemented by the digital filter, *D(z).* According to the equivalence specified by equation 9, and assuming a sampling period of T=0.001 seconds, the coefficients of the digital filter are:

$$\begin{aligned} A &= e^{-0.05} = 0.95 \\ B &= e^{-0.312} = 0.73 \\ C &= 4.0 \end{aligned} \tag{23}$$

Therefore, the equivalent digital filter is:

$$D(z) = 4.0\,\frac{z-0.95}{z-0.73} \tag{24}$$

The design procedure described above is very effective, and often produces low-cost high-performance motion control systems.

Although most digital motion control systems are designed for a specific application, designers often find it convenient to use general-purpose controllers which perform DC motor position control.

The digital control functions described in the preceding sections are available in the form of general-purpose controllers. The main advantage of general-purpose controllers is that they provide a complete design within days. General-purpose controllers are also the most economical solution when the number of required systems is moderate.

A typical general-purpose controller is the DMC-100 shown in Photo 1. This controller performs the digital control functions and allows you to vary the coefficients of the digital filter to tune the system for specific motor and load parameters. It can

TABLE 1

Parameter	Value	Definition
$K_t = 0.706$	Nm/A	Motor torque constant
r = 1.4	Ω	Motor resistance
L = 0	H	Motor inductance
$K_a = 5$	V/V	Amplifier gain
$V_s = \pm 10$	V	D/A converter output voltage
n = 8	Bits	Number of bits of D/A conversion
N = 500	Counts/revolution	Encoder resolution

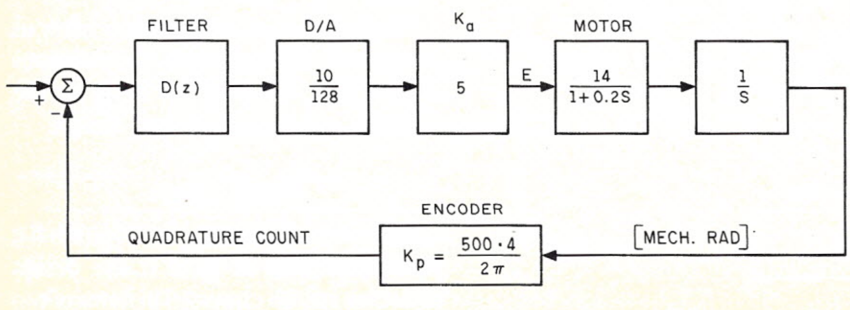


Figure 6. Specific system elements of the example digital motion control system.

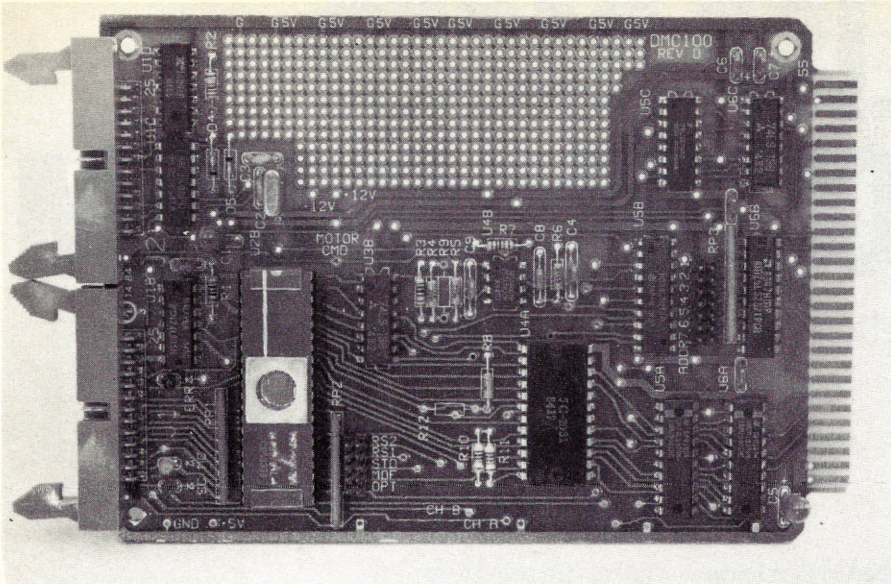


Photo 1. The DMC 100 Motion Controller. This controller closes the position loop, performs the digital filtering for stability, and controls the motion, distance, speed, and acceleration rates.

also generate a motion of specified distance with programmable acceleration and speed rates. This allows the designer to complete the motion control system design within hours.

High volume users may prefer to use IC-level controllers. Such a control system is available in the form of a chip set from Galil Motion Control. The chip set includes two ICs which perform the same control functions as the DMC 100 Controllers and can be integrated into the designer's circuits. The chip set (MCC 3000) is shown in Photo 2.

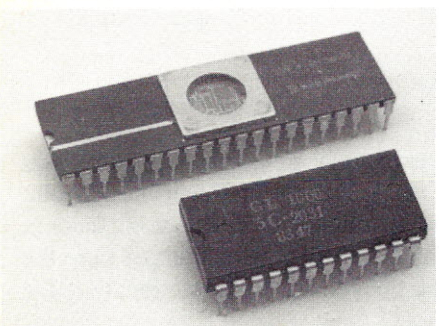


Photo 2. This chip set (MCC 3000) controls a DC motor and can be integrated into the designer circuit.

FUTURE TRENDS

Digital control systems of the type described here are becoming very common. The digital approach simplifies system design and improves system performance while reducing the cost. This digital control approach is expected to be used in most systems within the next five years.

Future digital control system designs will benefit more and more from the microprocessor's ability to perform advanced control algorithms. Such control algorithms include feedforward control which reduces the following error and cuts down the motion time.

The controllers with adaptive control are more advanced. Such controllers continuously estimate the system parameters and adjust the control parameters accordingly. Finally, a direct result of the parameter estimation is the fault detection capability. Future control systems will monitor the system parameters and report to the host computer when any system parameter exceeds the allowable range.

References

1. Tal, Jacob. *Motion Control by Microprocessors*. Mountan View, CA: Galil Motion Control, 1984.
2. Franklin, Gene F. and J. David Powell. *Digital Control of Dynamic Systems*. Addison-Wesley Publishing Co., 1980.
3. *DC Motors Speed Controls Servo Systems*. Hopkins, MN: Electro-Craft Corp.
4. Tal, Jacob and David Wolf. "Improved Design with General-Purpose Digital Motion Controller." Proceedings of Twelfth Annual Symposium, Incremental Motion Control Systems and Devices, Prof. B.C. Kuo, Editor. Champaign, IL, 1983.
5. Tal, Jacob. Workshop notes, Motion Control Seminars, Mountain View, CA.
6. Eveleigh, V.W. *Control Systems Design*. New York: McGraw-Hill Book Company. 1972.

Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

71	81	91
Excellent	Good	Fair

A LISP-BASED ROBOT CONTROL SYSTEM

Part II: Location and Object References

Doug Snead and John Roach
Department of Computer Science
Virginia Polytechnic Institute
Blacksburg, VA 24061

The simple robot control commands described in Part I provide very rudimentary robot arm control. A number of useful system additions come to mind. It would be nice to have some sort of a global list of places so that once a place is defined, you can simply refer to it by name and the associated joint vector is automatically retrieved. One way of doing this is to make an association list (or *a-list* for short). This a-list has a joint vector associated with a unique name. A typical place *a-list* might look like this:

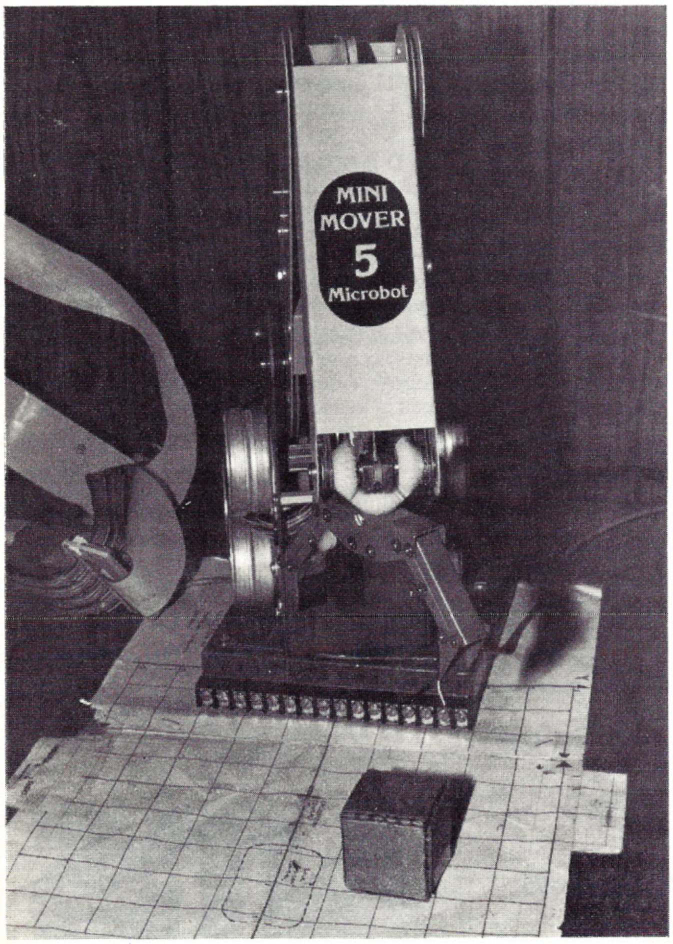


Photo 2. The Lisp commands described in this article can be used to move objects around with a robot arm. The robot commands allow you to reference objects and locations by name.

```
( (a (10 -100 1040 236 923 40))
  (overa (10 200 700 236 923 40))
  (b (500 200 300 900 800 50)) )
```

An a-list always has this general form. The first element is called the *name* or *key*. The rest of the elements are the data associated with the key. A special function called `assoc` (defined later) is used to retrieve a joint vector given its name.

Since we want the places to be accessible by all routines, we will make the a-list be the value of the global variable `*places`. We will also need functions to add new places and retrieve joint vectors for defined places. To define a place, the program requires you to manually set each arm joint using `setarm` and then requests a unique name to associate with that arm position.

```
(defun defplace ()
  (prog (aplace name)
    (msg '(enter a name for the place))
    (setq name (read))
    (msg '(position the arm))
    (setq aplace(setarm))                        ;position arm
    (setq *places
       (cons (list name aplace)
          *places))))
```

The function `defplace` prompts for a place name and then asks you to position the arm where you want it. The name and joint vector (place) are combined and added to the list of places in `*places`. Note that the value of `*here` must be changed since the changes and position must be remembered once you move the arm.

Now that you can add new places to `*places`, you need some way to retrieve them. The function `(place 'x)` returns the joint vector associated with `x`. If no such place exists, the function should report an error.

```
(defun place (placename)
  (prog (jointvec)
    (cond ((setq jointvec
         (car (cdr (assoc placenames
           *places))))
      (return jointvec))
      (t (error placename 'is 'not 'a
        'place)))))
```

Using an a-list simplifies the implementation since we can use a built-in Lisp function called `assoc` to search the a-list and return the associated information. With this method of stored vectors, you call the vectors by name using a command such as `(goarm (place 'X))`. As this is a little cumbersome, and you will use this command frequently, it is probably more convenient to imbed the place-to-vector conversion in other functions so that the above command could be written as `(goplace 'X)`. An example showing this type of arrangement is shown below.

```
(defun goplace (aplace)
  (goarm (place aplace)))
```

OBJECT DESCRIPTION

The `place` data type allows you to refer to places by name, but what if you want to have movable objects? How can objects be represented? Since a vision system is not available to provide object position descriptions, they must be represented by stored information. An object's name can be associated with the joint vector the arm must have in order to grip the object in a manner similar to the techniques used to implement `*places`. Thus, the starting point of such a scheme requires you to manually position the arm (and gripper) so that the object is gripped. Once an object's name and position are obtained, the position information must be updated whenever the object is moved.

The robot control system uses two additional global variables (`*objects` and `*holding`) to implement this feature. The global variable `*objects` contains an a-list of object names and their associated joint vectors. The global variable `*holding` is a flag indicating one of several different possible conditions for the concept of a gripper holding an object.

In the simplest case, the gripper is empty so we could let the value of `*holding` be `nil`. For the MiniMover-5,™ the one bit of feedback indicates when the gripper is grasping an object (within a certain tolerance) and that `*holding` should be non-nil. If the gripper is holding an object, we could let the value of `holding` be the name of the object. This approach requires each object to have a unique name before it can be used.

At this point we must deal with a difficult robot problem: object collision avoidance. Look at some of the things that must happen prior to the hand actually grasping the object. The arm can never move directly to the object's position; the gripper will probably knock the object over as it approaches. Instead, we need to define some approach to the object that will be unlikely to disturb other objects or knock over the desired one. The question is, how can this approach be determined given only the position of the arm and gripper when grasping the object? If the robot system had the capability to generate joint vectors from XYZ coordinates, then we could find an intermediate point somewhere above the object and move the arm to that point first. For example, if you had a chesspiece at XYZ location (100, 20, 8), you would first move the arm to location (100,20,40) and then to (100,20,8). This way the arm would not disturb the other pieces on the chessboard.

Since the present system does not support XYZ coordinates, we must find some other method of insuring a clear route. Several simplistic solutions are available. You may want to develop your own algorithms that include more complex path control computations.

The required computations can be simplified by restricting the environment to flat surfaces with objects that can be set down without having them roll away. One way to find a clear approach path in this situation is to have the user specify a clear approach. This path must be taken already since the user must move the gripper onto the object when its position is defined. Why not save the approach that is used?

At this point, we can tell what the a-list `*objects` should look like. `*Objects` is defined as a list of object names associated with a list of two joint vectors: one for the object's location and the other for the approach path (some arm position above the object). As an example, suppose you had two objects `block1` and `screw5`, then the value of `*objects` might be:

```
((block1 ((458 452 454 865 654 52) (458 552 484 865 654 252)))
 (screw5 ((258 753 951 842 268 25)
          (258 853 951 842 268 225))))
```

The robot control program requests three things when defining a new object: it needs to know the object's name, it needs to have the arm positioned at the approach path, it needs to have the arm positioned gripping the object. A function that does this and places the information in the `*objects` a-list is `defobject`.

```
(defun defobject ()
  (prog (objectname over on)
    (msg '(enter a name for the object))
    (setq objectname (read))                  ;user enters name
    (msg '(position the arm above the
      object))
```

```
(setq over (setarm))                    ;define approach
(msg '(grasp the object))
(setq on (setarm))                      ;grasp object
(cond ((eq (grip) 'empty) (setq on
(closearm))))
(setq *objects                          ;add to *objects
  (cons (list objectname
      (list on over))
    *objects))
(goarm (open over 150))))
```

The function closearm closes the claw until grip indicates something is gripped, returning the current arm joint position vector *here.

```
(defun closearm ()
  (progn
   pollclaw:
    (cond ((eq (grip) 'holding) *here)
     (t (goarm (addvec *here '(0 0 0 0 0 -10)))
       (go pollclaw:)))))
```

The defobject function defines objects if you can always manually describe an approach route. But what if you only know the object's position and don't want to manually define a place over it? Given our simplifying assumptions about the environment, another way might be to move the shoulder joint to lift the arm out of the way. The function (over X) computes the joint vector for some place over position X by adding a constant to the shoulder and elbow joints of position X.

```
(defun over (x)
  (addvec x
    '(0 -300 300 0 0 0)))               ;add to first joint
```

Now, an approach can be generated even if you only know the object position. At this point we can write a grasp function that only needs the object name to determine how to pick up the object. When you want to pick up an object, you simply need to say something like (grasp 'box7) and the arm will move to above box7, open the gripper, descend to box7, and close the gripper. Once the object is grasped, the variable *holding contains the object's name. Function grasp should always check the value of *holding before taking any action. This will prevent a program from attempting to pick up an object when another object is already being held. The grasp function can be defined as follows:

```
(defun grasp (obj)
  (prog (onover onobj overobj)
   (cond (*holding
     (error '(grasp fails) 'holding
       *holding)))
   (setq onover (getapproach obj))
   (setq onobj (car onover))
   (setq overobj (car (cdr onover)))
   (goarm (open overobj 150))
   (goarm (open onobj 150))
   (goarm onobj)                        ;obj in gripper
   (cond ((eq (grip) 'empty)
      (error '(grasp fails) obj 'not
        'there)))
   (setq *holding obj) ))               ;update *holding
```

Function grasp uses the function getapproach to set onobj (the objects position) and overobj (somewhere over onobj) from the information stored in *objects. If no overobj position is found, then the function (over onobj) is used to generate one.

```
(defun getapproach (object) (jointvecs)
 (setq jointvecs
  (car (cdr (assoc object *objects))))      ;get obj info
 (cond ((null jointvecs)
  (error 'undefined 'object object))
  ((null (car (cdr jointvecs)))              ;no overobj
  (return (list (car jointvecs)
     (over (car jointvecs)))))               ;make one
  (t (return jointvecs)))))
```

Function open is used to add a value to the gripper component of a joint vector. The function call (open jointvec x) returns a new joint vector, the same as jointvec except x is added to the sixth component.

```
(defun open (jointvec x)
  (addvec jointvec (list 0 0 0 0 0 x)))

(defun claw (place)                          ;return 6th compo-
                                              nent of joint
                                              vector
  (car (cdr (cdr (cdr (cdr (cdr
   place)))))))
```

Now that you can grasp objects, you can also move them around. Moving objects from place to place requires that the gripper be continually clamped about the object. Function goarm must be modified to insure that the gripper is not opened when variable *holding is non-nil. When *holding is non-nil, the gripper position should not be changed.

```
(defun goarm (jointvec)
 (prog (here2)
  (cond
   (*holding                                 ;something in claw?
   (steparm (subvec (setq here2
     (open jointvec
      (sub (claw *here)
       (claw jointvec))))
     *here))                                 ;don't change
                                              gripper position
   (setq *here here2)
   (cond ((eq (grip) 'empty)                 ;check for object
     (error '(goarm reports object
      dropped)))))
   (t (cond ((eq (steparm
     (subvec jointvec *here)) t)
     (setq *here jointvec))
     (t (error '(steparm fails))))))
  (return t)))
```

Because a function is needed to update an object's position when it is moved, we also need a function to remove an object and its associated information from the *objects a-list.

```
(defun remove-el (element a-list)
 (cond ((null a-list) nil)                   ;element not here
  ((eq (car (car a-list))
```

U.S. Postal Service

STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION

Required by 39 U.S.C. 3685)

1A. TITLE OF PUBLICATION	1B. PUBLICATION NO.	2. DATE OF FILING
Robotics Age	0 1 9 7 1 9 0 5	10/1/84
3. FREQUENCY OF ISSUE	**3A. NO. OF ISSUES PUBLISHED ANNUALLY**	**3B. ANNUAL SUBSCRIPTION PRICE**
Monthly	12	$24.00

4. COMPLETE MAILING ADDRESS OF KNOWN OFFICE OF PUBLICATION *(Street, City, County, State and ZIP Code) (Not printers)*

Strand Building, 174 Concord Street, Peterborough, NH 03458

5. COMPLETE MAILING ADDRESS OF THE HEADQUARTERS OF GENERAL BUSINESS OFFICES OF THE PUBLISHER *(Not printer)*

Strand Building, 174 Concord Street, Peterborough, NH 03458

6. FULL NAMES AND COMPLETE MAILING ADDRESS OF PUBLISHER, EDITOR, AND MANAGING EDITOR *(This item MUST NOT be blank)*

PUBLISHER *(Name and Complete Mailing Address)*

Carl T. Helmers, Jr., Strand Building, 174 Concord St., Peterborough, NH 03458

EDITOR *(Name and Complete Mailing Address)*

Raymond GA Cote, Strand Building, 174 Concord St., Peterborough, NH 03458

MANAGING EDITOR *(Name and Complete Mailing Address)*

7. OWNER *(If owned by a corporation, its name and address must be stated and also immediately thereunder the names and addresses of stockholders owning or holding 1 percent or more of total amount of stock. If not owned by a corporation, the names and addresses of the individual owners must be given. If owned by a partnership or other unincorporated firm, its name and address, as well as that of each individual must be given. If the publication is published by a nonprofit organization, its name and address must be stated.) (Item must be completed.)*

Full Name: Robotics Age, Inc., 174 Concord St., Strand Bldg., Peterborough, NH 03458

Stockholders owning more than 1% of Robotics Age, Inc.
North American Technology, Inc., 174 Concord St., Peterborough, NH 03458
Robotics Publishing Corp., PO Box 720, Conroe, TX 77305

Stockholders owning more than 1% of North American Technology, Inc.:
Carl T. Helmers, Jr., 174 Concord St., Strand Bldg., Peterborough, NH 03458

Stockholders owning more than 1% of Robotics Publishing Corp:
Philip C. Flora, P.O. Box 720, Conroe, TX 77305
Alan R. Thompson, P.O. Box 720, Conroe, TX 77305

9. FOR COMPLETION BY NONPROFIT ORGANIZATIONS AUTHORIZED TO MAIL AT SPECIAL RATES *(Section 423.12 DMM only)*
The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes *(Check one)*

☐ (1) HAS NOT CHANGED DURING PRECEDING 12 MONTHS ☐ (2) HAS CHANGED DURING PRECEDING 12 MONTHS *(If changed, publisher must submit explanation of change with this statement.)*

10. EXTENT AND NATURE OF CIRCULATION	AVERAGE NO. COPIES EACH ISSUE DURING PRECEDING 12 MONTHS	ACTUAL NO. COPIES OF SINGLE ISSUE PUBLISHED NEAREST TO FILING DATE
A. TOTAL NO. COPIES *(Net Press Run)*	21,867	20,045
B. PAID CIRCULATION 1. Sales through dealers and carriers, street vendors and counter sales	10,047	10,040
2. Mail Subscription	10,121	9,917
C. TOTAL PAID CIRCULATION *(Sum of 10B1 and 10B2)*	20,168	19,957
D. FREE DISTRIBUTION BY MAIL, CARRIER OR OTHER MEANS SAMPLES, COMPLIMENTARY, AND OTHER FREE COPIES	93	35
E. TOTAL DISTRIBUTION *(Sum of C and D)*	20,261	19,992
F. COPIES NOT DISTRIBUTED 1. Office use, left over, unaccounted, spoiled after printing	108	31
2. Return from News Agents	1,498	22
G. TOTAL *(Sum of E, F1 and 2—should equal net press run shown in A)*	21,867	20,045

11. I certify that the statements made by me above are correct and complete	SIGNATURE AND TITLE OF EDITOR, PUBLISHER, BUSINESS MANAGER, OR OWNER: Carl T. Helmers, Jr.

```
      element)                          ;remove element
    (cdr a-list))
    (t (cons (car a-list)
      (remove-el element                ;repeat
        (cdr a-list))))))
```

Since function `remove-el` just returns a new copy of a-list, the following functions are required to permanently change `*objects` or `*places`.

```
(defun remobject (object)
  (setq *objects
    (remove-el object *objects)))


(defun remplace (place)
  (setq *places
    (remove-el place *places)))
```

Now that you can move objects without dropping them, it would be useful to somehow release them. When you release an object you must remember both the place it was released and the place over the release point (the approach path).

```
(defun release ()                       ;release object
  (cond ((null *holding)
    (error '(releasing fails-nothing
      held)))
    (t (remobject *holding)             ;clear old info
      (setq *objects
        (cons (list *holding
          (list *here
            (over *here))))
      (setq *holding nil)
      (goarm (open *here 150)))))
```

The command to move an object to a particular place now appears relatively trivial. All of the complicated work is being done by the lower-level functions. Given a place `p` and an object `o`, the command `(move o p)` moves object `o` from its current position to position `p`.

```
(defun move (obj aplace)
  (prog (placevec)
    (setq placevec (place aplace))
    (grasp obj)                         ;get object
    (goarm (over *here))                ;lift object
    (goarm (over placevec))             ;above aplace
    (goarm placevec)                    ;position object
    (release)                           ;now let go
    (goarm (over placevec))))           ;lift arm
```

This function is useful for moving pieces around board games such as checkers, monopoly, and chess. You would first need to define places for each possible move on the board and then define the gamepieces as objects.

USING TWO ARMS

It is often necessary to use more than one arm for a single task. The MiniMover-5 allows two arms to be connected in parallel through the Apple interface. Unfortunately, both arms cannot be moved at the same time. An arm can be selected with the BASIC command `@arm n` where `n` is either 1 or 2. If the `@arm`

command is never issued, then arm 1 is assumed. The arm selection function in Lisp is quite similar.

```
(defun selectarm (n)                    ;1 or 2 only
  (progn
    (prin1 '@arm)
    (print n)))
```

The function (selectarm n) allows you to select either arm1 or arm2. One small problem arises. Every time the (pr# *armslot) command is issued (as in robot-rep), the internal arm pointer (1 or 2) is initialized to 1. This problem can be remedied by redefining (robot-rep) and (msg) to issue the (selectarm *arm) command after every (pr# *armslot).

```
(defun robot-rep ()
  (prog (rcmmd rcval)
    loop:
      (print '(enter robot command))
      (setq rcmmd (read))               ;get command
      (pr# *armslot)
      (selectarm *arm)
      (setq rcval (eval rcmmd))         ;eval it
      (pr# 0)
      (print rcval)                     ;print return
                                         value

      (cond ((eq rcval 'exit)
        (return 'robot-rep))
        ((eq rcval 'init) (arminit)))
    (go loop:)))

(defun msg fexpr (message)
  (progn
    (pr# 0)
    (print (eval (cons 'list message)))
    (pr# *armslot)
    (selectarm *arm)))

(defun grip ()                          ;modify grip to
  (cond ((> (port *arm *armslot 7) 128)
   'empty)                              ;include *arm
    (t 'holding)))
```

These definition changes would work fine, since you could define places and objects for both arms, but a logical inconsistency appears. Places and objects defined for one arm would have no real meaning for the other arm. In practice, you never want to use places defined for one arm with the other arm since each place is defined as a specific arm's joint angles. A place defined for one arm but used by the other would just copy the first arm's joint positions for that place. An object defined for one arm and used by the other would result in grasping at empty air. In addition, since object and place names must be unique, the same name for a single object could not be used by both arms.

One way out of this problem for two arms is to keep separate global arm variables for each arm. This means that we must have two copies of *here, *places, *objects, and *holding. Every time control is switched from one arm to the other, the global variables are also switched. A global variable *arm is added to the system to remember which arm is current, 1 or 2. Since the arm number

is initially 1, the function (robotsysinit) takes care of initializing *arm.

```
(defun robotsysinit () (seq *arm 1))
```

To avoid having to distinguish between arms in the functions defined so far, we can add the global variables *here1, *here2, *places1, *places2, etc. The values of *here, *places, *objects, and *holding is updated every time the arms are switched. The functions (arm1) and (arm2) do this switching.

```
(defun arm1()                              ;switch to control
                                            arm 1
 (progn
  (cond ((eq *arm 1) (error '(already
  arm1))))
  (setq *here2 *here)
  (setq *places2 *places)
  (setq *objects2 *objects)
  (setq *holding2 *holding)
  (setq *arm 1)
  (setq *here *here1)
  (setq *places *places1)
  (setq *objects *objects1)
  (setq *holding *holdings1)
  (selectarm 1)))
(defun arm2()                              ;switch to control
                                            arm 2
 (progn
  (cond ((eq *arm 2) (error '(already
  arm2))))
  (setq *here1 *here)
  (setq *places1 *places)
  (setq *objects1 *objects)
  (setq *holding1 *holding)
  (setq *arm 2)
  (setq *here *here2)
  (setq *places *places2)
  (setq *objects *objects2)
  (setq *holding *holdings2)
  (selectarm 2)))
```

Now you can define objects and places for both arms and not worry about inadvertently redefining a place or object for the other arm.

NEXT MONTH

The commands described in Parts I and II provide a useful, flexible robot control system. Next month, Part III describes some actual operating experiences with this control system.

MiniMover-5 is a registered trademark of Microbot, Inc. Apple II is a registered trademark of Apple Computer Corp.

Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

72	82	92
Excellent	Good	Fair

Advertising

Reader Service

R.S. No. Company

1 Advanced Micro Systems, p. 11
2 Amarobot, Cov. II
3 AMSI Corp., p. 6
4 Arctec Systems, Cover III
5 AstroTronics Microsystems, p. 15
6 Berger Lahr Corp., p. 35
7 Bersearch Info. Services, p. 5
8 Centroid, p. 33
9 Crabapple Systems, p. 7
10 Cybernetic Micro Systems, p. 13
11 CyberPak Co., p. 4
12 Datacraft, p. 31
13 Digi-Tech, Inc., p. 32
14 Feedback, Inc., p. 2
15 HoB-Bots of Hudson, p. 32
16 Laboratory Microsystems, Inc., p. 7
17 Lab-Volt, p. 3
18 Light Machines, p. 16
19 MicroMint, Inc., p. 14
20 Micron Technology, Inc., p. 15
21 Northwest Computer Algorithms, p. 17
22 Oriental Motor USA Corp., p. 17
23 Personal Robotics Corp., p. 27
24 Prism Robotics, Inc., p. 23
25 Rio Grande Robotics, p. 31
* Robotics Age Advertising, p. 18
* Robotics Age Subscriptions, Insert
* Robotics Age Back Issues, Insert
26 Robotronix, Inc., p. 33
27 Schulz Enterprises, Inc., p. 31
29 Servo Dynamics, p. 18
28 Small Parts, Inc., p. 14
30 Stock Drive Products, p. 5
31 Superior Electric Co., Cov. IV
32 Valley Robotics, p. 32
33 Vantec, p. 32
34 Vernitech, p. 31

**Correspond directly with company*

ROBOTIX

Part I: Examining the Pieces

Mark J. Robillard
MJR Digital
PO Box 630
Townsend, MA 01469
Source ID: TI7851

Two years ago, I wrote a series of articles for *Robotics Age* that described the building of several hand, arm, and mobile robot systems. The final article in the series presented a review of a revolutionary product in the field of hobby robotics. This product, the Armatron™ (then by Tomy Toys, most recently available through Radio Shack stores), has become a standard among experimenters for low-cost arms. Since then, there have been many developments in the industry. Heath Company introduced HERO I,™ RB Robot Corp. introduced the RB5X,™ and Androbot introduced a series of robots from Topo™ to BOB.™

All these announcements would lead you to believe that the systems end of hobby robotics is becoming the norm. You purchase a complete kit or completely assembled robot. What about those of you who are just interested in building arms? Or maybe there are a few left who enjoy building the entire system?

The other day, while walking through a local department store, I came upon a new product. There on the shelf was a package with a familiar sounding name—*Robotix*.™ Robotix is a Lego™-type building kit designed for children ages 7 to 14. The product is available as two kits; Models R1000 and R2000. Although basically identical, the R2000 kit contains more parts and two more control motors.

What can you build with a Milton Bradley Robotix kit? In less than two minutes, you can build an arm with the same functionality, yet a slightly stronger grip, as the Armatron arm. You can then break that arm down and make a four-wheeled vehicle with an arm that can rove around your house. You can also build two arms on a smaller vehicle.

Photo 1. The two, new Robotix series kits from Milton Bradley.

Sounds tremendous doesn't it? Well, you can't do all this with just *one* kit. There are limitations. The limitations are in the number of motors and in the number of structural parts.

PARTS

Photo 2 shows the parts supplied in the smaller R1000 kit. Each R1000 kit contains two motors. This is a serious limitation and you will find that you'll want to buy more than one kit. The retail price at my local department store for the R1000 kit is $39.95. The R2000 kit is $59.95, which is substantially more, but you get two extra motors and a larger control box.

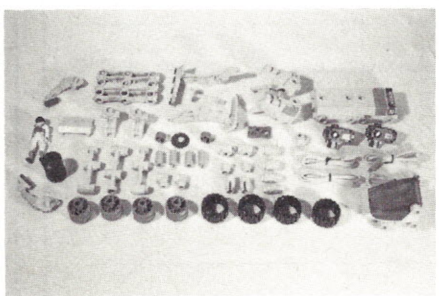


Photo 2. A layout of parts provided in the Robotix R1000 kit.

The R1000 kit is designed for constructing mobile robots with self-contained battery supplies. Photo 3 shows the R1000 control box. This box contains two control switches and space for four C-cell batteries. Four two-pronged connectors are on the front. These are electrical connections for powering the system motors. The box allows you to control two motors independently.

The reason I say that this kit is meant for mobile robotics is that the battery box control switch assembly can be attached to the robot. A typical vechicle, such as the walking vehicle shown in Photo 4, has the battery/switch assembly mounted in the rear section of the assembly. (By the way, the small astronaut is included in the kit.) Let's examine this vehicle piece by piece.

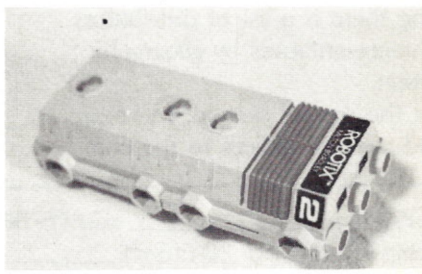


Photo 3. The Robotix R1000 battery and switch control box.

Structural Elements. The main structural element is a shaft-like assembly with many outlets, inlets, and fastening areas. Photo 5 provides a close-up view of this piece. You can connect pieces to this structural member in many different ways. This particular shaft assembly can become an axle, a support, or a structural member in an arm. It's made of rather thick styrene plastic and has an octagonal connection structure.

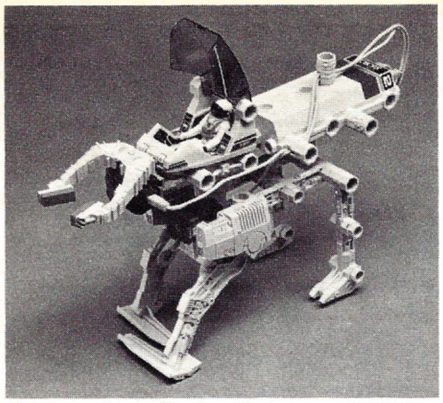


Photo 4. The complete walking rover vehicle fully equipped with gripper mechanism. The battery and control box are mounted on the end of the vehicle.

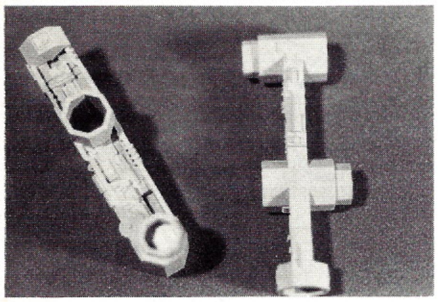


Photo 5. Main structural support member for the walking vechicle.

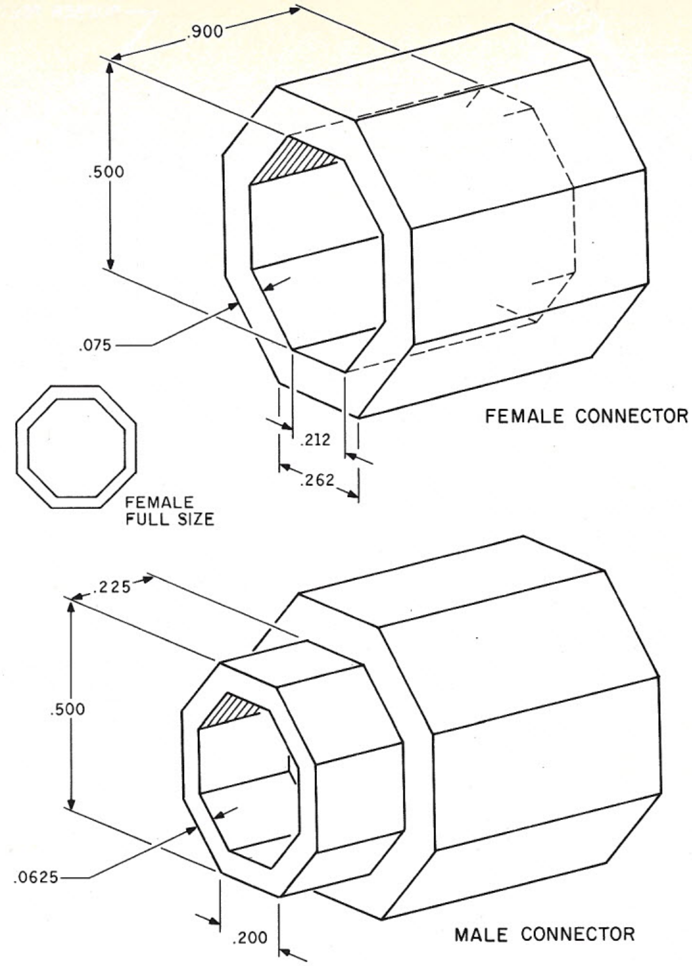


Figure 1. Mechanical details of the octagonal connector element used throughout the Robotix series structural pieces. All dimensions are in inches.

This structure is detailed in Figure 1. This drawing is provided in case you wish to augment the kit parts with your own parts built of wood, plastic, or light metals. To connect the various assemblies within the Robotix kits, you need to know the details of this connection scheme. All the measurements shown in Figure 1 were taken with a pair of calipers so they should be accurate. However, some sanding may be necessary since I do not know how tightly tolerances are held on a consumer toy.

Let's look at some of the other structural pieces. Photo 6 shows a selection of L-shaped pieces for corners, T-shaped pieces, and what I call in-line connectors that have two male ends. Other in-line connectors have a male and a female end. One particularly interesting piece is called the centering jig which mounts on the back of a motor and provides the octagonal connection socket exactly centered on the motor. This is necessary for such things as grippers that must be centered on the end of an arm. Only one of these centering devices is provided with each kit.

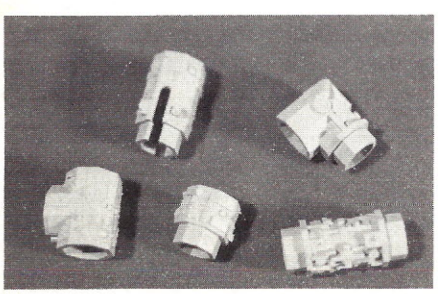


Photo 6. Other structural support pieces provided in the R1000 and R2000 kits.

Hands and Legs. I'm sure anyone familiar with the Armatron gripper arrangement and the HERO I gripper device will join me in dismay at the gripping force (or lack of such) that they display. The Robotix gripper arrangement shown in Photo 7 is a two-fingered system consisting of two structural pieces of plastic. A special type of foam rubber is mounted on the end of each finger. This rubber end has the feel of human skin but the consistency of hard rubber. Although it does have some resilience, it has the feel of a no-slip surface.

The gripper assembly is quite ingenious. The motor assembly design allows the grippers to be assembled in less than ten seconds. Figure 2 shows how the gripper is assembled. A pin arrangement used to pivot one side is built into the motor assembly. This greatly simplifies construction. As you can see, just moving the three pieces together and pressing them down on the octagonal connector produces an instant gripper!

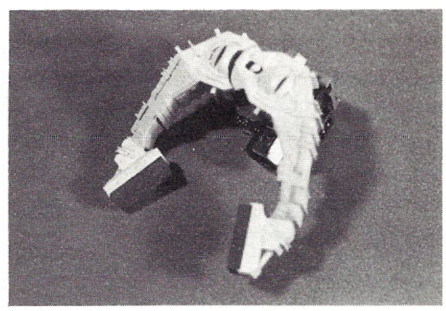


Photo 7. Close-up of the assembled gripper mechanism. Note the foam rubber pads at the finger tips.

The two-legged walker assembly is not quite as simple. This assembly might take you 30 seconds to put together and then might take even a few more seconds to make it work. The pieces are shown in

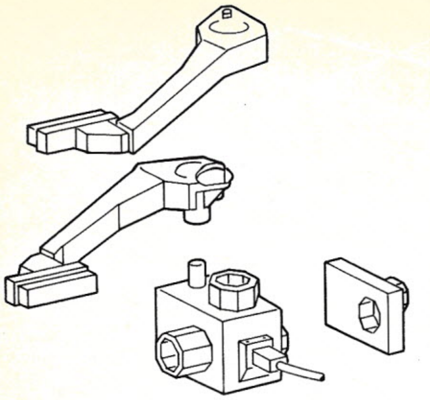


Figure 2. Assembly sequence detail for a Robotix series gripper.

Photo 8 and detailed in Figure 3. In my experience, the legs move too slowly. I've had problems with one kit binding so much that it actually stopped the walker. Some further exprimentation in this area is warranted. I have not tried the legs in another kit to see if it was just one particular kit that worked improperly.

Power. I have yet to kill a set of batteries, even after many hours of use. This says much for the motors used in the kit. Photo 9 shows a complete motor assembly. The little labels on the side indicate the motor number for control purposes. The motor assembly is a small black structure with multiple octagonal connector areas. One of those octagonal connectors is the actual drive shaft. It only comes out on one

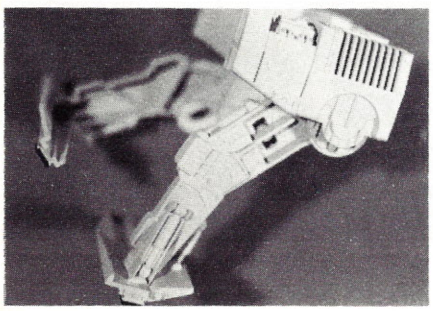


Photo 8. Assembled walker structure. Foam rubber pads are also included for the bottoms of the feet.

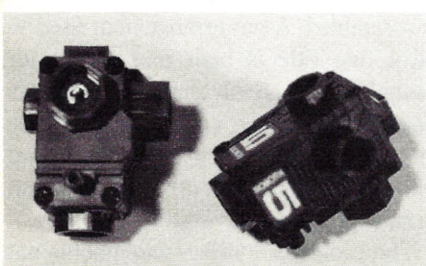


Photo 9. Close-up view of the Robotix series motor. The drive hub is shown in the upper left-hand corner. The guide pin described in the text is located beneath the drive hub in the photo.

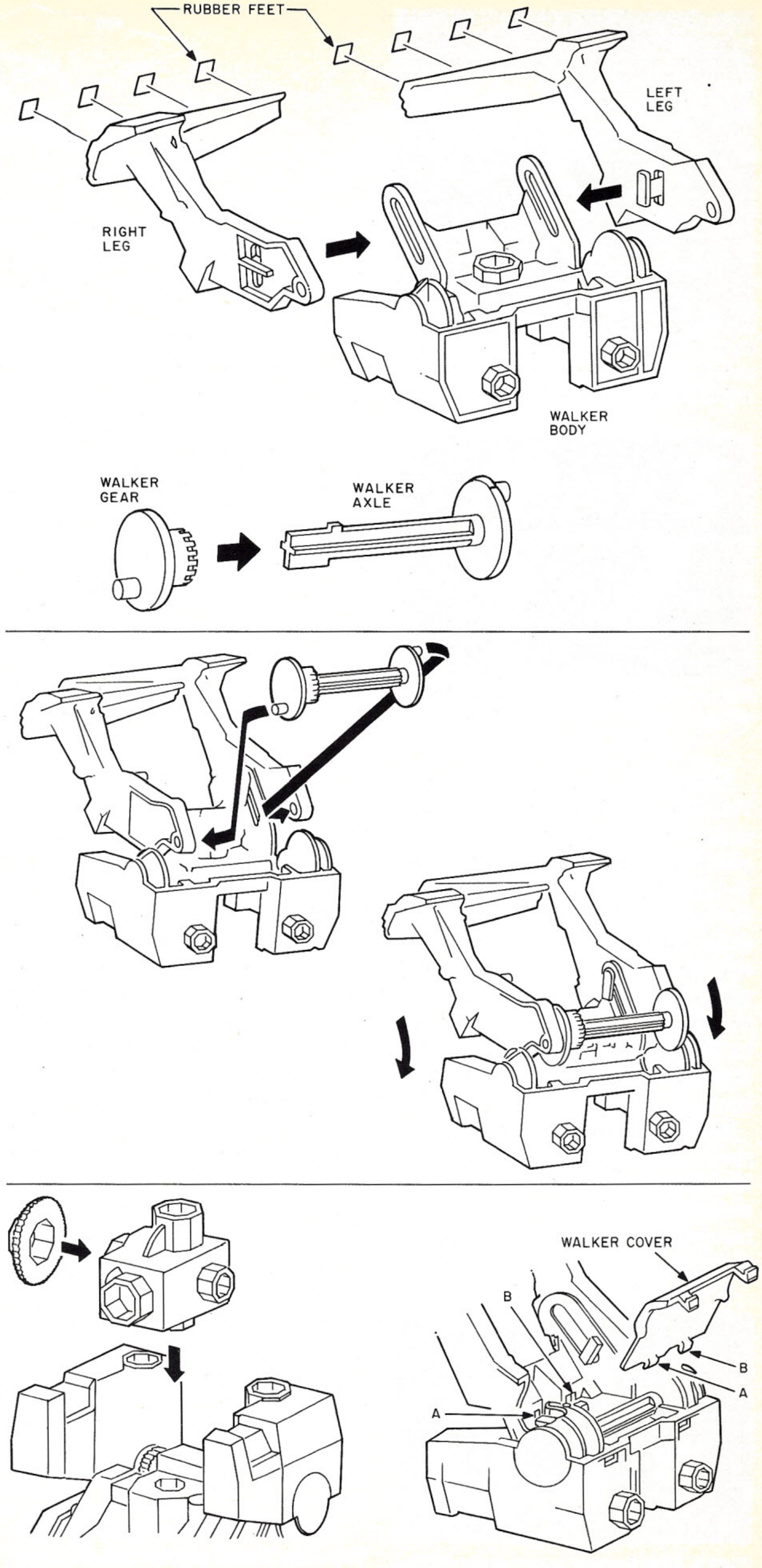


Figure 3. Assembly drawing showing the various pieces and assembly sequence for Robotix leg structure.

side (shown facing up in Photo 6) and turns at 6 rpm. This is a 6 to 1 reduction from the internal motor speed. I am told that this small 3 V motor is so geared down that it is capable of generating a torque of 50 ounce-inches; quite substantial for a motor of this size.

The motors with which I have experimented draw approximately 70 mA under a no-load condition. Under a full load stall condition, they typically draw around 190 mA although some draw over 200 mA. The electrical connection is a two-pronged plug similar to the one on the main control box. A simple cable is provided to connect the motors to the controller. The motors are so light weight, they add very little to the weight of an assembled arm.

Up to this point I have only talked about the R1000 series control box. Photo 10 shows the R2000 series control box. The R2000 switch box contains no batteries. The batteries are in a separate, larger assembly. Because there are four motors in the R2000 series, Milton Bradley has decided to power these with D-cell batteries.

The R2000 battery box (shown in Photo 11) is a large assembly that can be used as a complete base for construction projects. The five-position switch assembly shown in Photo 10 and the battery box in Photo 11 are connected with a three-conductor cable. Three wires are used since the 6 V inside the battery box is divided into a ± 3 V supply with a common line.

Each of the five switches on the control box provides either $+3$ V or -3 V to one leg of the motor. The other end of the motor is connected to a common ground. This arrangement is much simpler than switching the direction of current and requires a less expensive switch. Each one of the five switches is a momentary action in each direction. The R2000 switches are normally *off*. The R1000 switches are either *on* in one direction or *off* in the other direction, making the R1000 controller more appropriate for mobile robotics—you can just turn the machine on and let it go.

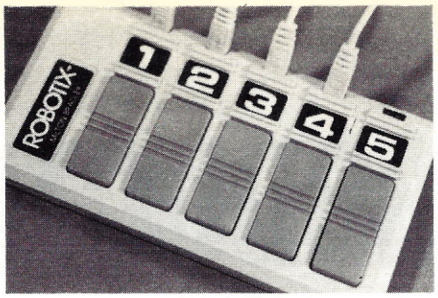


Photo 10. The remote switch box provided in the Robotix R2000 kit. Also shown are the cables plugged into the control sockets.

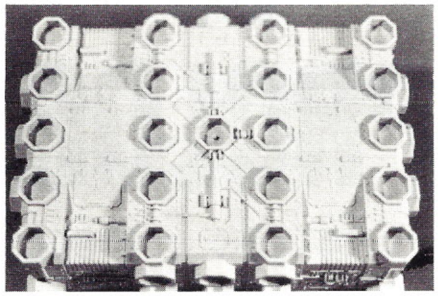


Photo 11. This large structure contains the four D-cell batteries for powering the R2000 kit. It can also be used as a base support for many structures.

NEXT MONTH

Well, what have we done so far? I've introduced the Robotix series, both the R1000 and R2000 kits. You know the approximate price. You know you can get them at department stores (they'll probably be in the Lego aisle or the build-it kit aisle with the Erector™ sets). You know that Robotix comes with a series of motors, structural parts, and switches. Each set contains one gripper assembly. How can you go wrong?

What I haven't provided so far are instructions for connecting the motors to your favorite home computer system and creating intelligent assemblies. In part 2, I will describe how the Robotix kit can be controlled from an external source.

I would like to extend my sincere thanks to the following people at Milton Bradley for their support. Mr. George Merritt, vice president, Public Relations, for putting me in touch with the correct people within the company. Mr. Gary Burgmann, product manager, Robotix Products, for the technical specifications and information regarding the manufacture and future of the series. Ms. Barbara Gemme, Secretary to George Merritt, for doing an excellent job of providing photographs and orchestrating contacts between everyone involved.

Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

73	83	93
Excellent	Good	Fair

INTRODUCTION TO PROLOG

Part II: A Simple Expert System

Dean A. Schlobohm
Micro-AI
PO Box 91
Rheem Valley, CA 94570

Although you have learned only the very basic Prolog commands, it is still sufficient to write a very simple *expert* system. Since the program in Listing 1 (see pages 17, 18 in the November issue) uses a number of Prolog procedures that have not been discussed in the article, the interested reader should consult one of the books on Prolog for further explanation.

There are many *expert* systems that solve problems by using a set of simple rules. Each of these rules is called an *if-then* rule, a *situation-action* rule, or a *production rule*. The purpose of this section is to present an example of how production rules can be implemented in Prolog to produce a simple expert system.

Experience has shown that rules of the following form can be used by a computer to solve problems that, if done by a human, would be thought of as exhibiting a certain degree of intelligence. The basic form of the rules is

IF (fact1 is true) AND
  (fact2 is true) AND
  .
  .
  (factn is true)
THEN (conclusion1 is true).

For example, the following can be thought of as production rules:

IF (a tree is green in winter)
THEN (it is an evergreen).

IF (an animal is a mammal) AND
  (the animal swims) AND
  (the animal is gray) AND
  (the animal is huge)
THEN (the animal is a whale).

IF (the infection is a pelvic-abcess) AND
  (there are rules that mention in their premise Enterobacteriacae) AND
  (there are rules that mention in their premise gram positive rods)
THEN (there is suggestive evidence (0.4) that the rules dealing with Enterobacteriaceae should be evoked before those dealing with gram positive rods).

Systems based on the above type of rules have produced convincing results in such diverse fields as medical diagnosis (MYCIN), chemical analysis (DENDRAL) and mineral exploration (PROSPECTOR). In the simple "expert" system discussed here, we will consider rules that allow the computer to guess an animal you are thinking of.

In Prolog, each production rule of the form:

IF (fact1 is true) AND
  (fact2 is true) AND...
  (factn is true)
THEN (conclusion is true).

can be expressed by a Prolog rule of the form

```
conclusion :-
 fact1,
 fact2, . . .
 factn.
```

Thus, to express the rule:

IF (an animal has hair)
THEN (it is a mammal).

you use the Prolog rule

```
isa(Animal,mammal) :-
 has(Animal, hair),  % Rule 1
 asserta(isa(Animal,mammal)).
```

Here the goal `isa(Animal,mammal)` means that `Animal` is a `mammal`. Similarly, `has(Animal,hair)` means that `Animal` has hair. Thus, the above Prolog rule states that `Animal` is a mammal if it has hair.

Similarly, you can express the more complex rule

IF (an animal is a mammal) AND
  (the animal is a carnivore) AND
  (the animal has black stripes)
THEN (the animal is a tiger).

by the Prolog clause

```
isa(Animal,tiger) :-
 isa(Animal,mammal),
 isa(Animal,carnivore),
 isa(Animal,'black
 stripes'),  %Rule 2
 asserta(isa(Animal,tiger)).
```

Finally, you can add facts about a given situation to your database. For example, if you know that animal1 has hair, is a carnivore, and has black stripes, then you add the following clauses to your database

```
has (animal1,hair). % Fact 1
isa(animal1,carnivore). % Fact 2
has(animal1,'black   stripes').
%Fact 3
```

If you then ask isa(animal1,mammal)? Prolog uses Rule 1 and Fact 1 to answer **yes. Similarly, if you want to know what animal1 is, you enter isa(animal1,X)? Prolog uses Rule 2, Rule 1, and Facts 1, 2, and 3 to answer X = tiger.

As you can see, Prolog is a convenient language for implementing a production rule system.

While this "expert" system illustrates how a computer could achieve a low level of problem-solving ability, most humans will probably want to know how the computer arrived at its conclusion. For example, if a program performed medical diagnoses, the doctor would probably want to know how the computer arrived at its diagnosis of the disease. In fact, the doctor might be open to a malpractice suit if he or she simply relied on the computer. Thus, any useful expert system should allow the user to ask the computer to explain how it reached its conclusion.

In general, the issue of how to have the computer generate a useful explanation is complex. For example, if the explanation is too simple, it may not be useful. Whereas if the explanation is not detailed enough, it may not be understandable.

Our simple system avoids these problems by having each rule contain a simple description of itself. These descriptions are then kept in a list which is passed to each rule. If a rule succeeds, then the rule description is added to the list. For example, Rule 1 can be rewritten as

```
isa(Animal,mammal,How) :-
 concat(['RULE 1: ',Animal,' is a
 mammal if it has hair.'],Z),
 has(Animal,hair,W),
 append([Z],W,How),
 asserta(isa(Animal,mam-
 mal,How)).
```

Here, the argument How is the list of rule explanations. When a fact is added to the database, an explanation is alse added. For example, if animal1 has hair, this fact is expressed as has(animal1,hair,'FACT: animal1 has hair').

An explanation can be generated simply by printing the list of facts and rules. This is done by the procedure how in Listing 1. For example, if the computer discovered that animal1 is a tiger, the explanation is

```
RULE 2: animal1 is a tiger if it is
  a mammal, a carnivore, and has
  black stripes.
RULE 1: animal1 is a mammal if it
   has hair.
FACT: animal1 has hair.
RULE 3: animal1 is a carnivore if
  it eats meat.
FACT: animal1 eats meat.
FACT: animal1 has black stripes.
```

To make the system even more useful, you should not be required to add all the facts you know about the situation in question to the database before running the program. In particular, the computer should simply ask you for more facts only as it needs them. For instance, if the computer is attempting to determine whether an animal is a mammal, it should ask the user if the animal has hair, if it does not know the fact already. If the program is written to ask the user for information which it needs, the user may want to ask the computer why it needs to know the fact. This is also extremely useful in debugging the program.

The system discussed here provides a simple mechanism for asking the user only the necessary facts and permitting the user to ask the computer why a particular fact is needed. This is accomplished by adding another argument to each rule definition. This argument is a list of the rules which have been used up to the point the user is asked for more data. For example, if the program asks Does animal1 have hair? you could enter why to request the reason the information is required. In the example, the program would respond with

```
RULE 2: animal1 is a tiger if it is
a mammal, a carnivore, and has
black stripes.
```

If this answer does not satisfy you, ask why again. The program responds with

```
RULE 1: animal1 is a mammal if it
 has hair.
```

At this point you know why the question was asked. The procedures ask_about and respond in Listing 1 show examples of how to implement *why* in Prolog.

Listing 1 shows the simple expert Animals system which attempts to determine what type of an animal you are thinking of. The program asks you questions to determine the animal in question. You answer either yes, no, or why to each question. Answer why if you want to know why the program asked you for certain information. After the program has reached its conclusion, it asks you if you want to have the program explain its reasoning. If you enter yes the program prints the explanation.

BIBLIOGRAPHY

Periodicals

Buchanan, B.G., and E. A. Feigenbaum. "DENDRAL and Meta-DENDRAL: Their applications dimension." *Journal of Artificial Intelligence* 11, (1978):5–24.

Clark, K. L. and F. G. McCabe. "PROLOG: a language for implementing expert systems." *Machine Intelligence* 10 (1982):455–475.

Duda, R.O., and P.E. Hart, N.J. Nilsson, R. Reboh, J. Slocum, and G. Sutherland. "Development of the PROSPECTOR consultation system for mineral exploration." SRI Projects 5821 and 6415, SRI International, Stanford, 1978.

Pereira and Warren. "Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks." *Artificial Intelligence,* 13:3(1980):231–278.

Books

Clocksin, W.F., and C.S. Mellish. *Programming in Prolog.* New York: Springer-Verlag, 1981.

Shortliffe, E.H.. *Computer-based medical consultations: Mycin.* New York: American Elsevier, 1976.

The version of Prolog discussed in this article is Prolog-86, an implementation of Prolog currently available for microcomputers running the MS-DOS and CP/M-86 operating systems. Prolog-86™ is a trademark of Micro-AI. MS-DOS™ is a trademark of Microsoft. CP/M-86™ is a trademark of Digital Research.

Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

74	84	94
Excellent	Good	Fair

A THIRD-GENERATION STEPPER MOTOR CONTROLLER

Part I: Intelligent Control

Ed Klingman and Karen Moty
Cybernetic Micro Systems
PO Box 3000
San Gregorio, CA 94074

Stepper motor controllers have undergone a series of refinements and advancements. The first stepper motor controllers consisted of bulky boxes with switches that set the number of steps and stepping direction and buttons to actually start the motion. Step rates were set in hardware, as were the acceleration and deceleration characteristics. These early controllers were obviously meant for manual operation. They were expensive, heavy, and large compared to the motors being controlled.

The next controller design saw the box functions condensed onto a single printed circuit board. This significantly reduced the cost and packaging requirements, but did not increase the controller capabilities. One additional design benefit was the ability to electronically simulate switch inputs, allowing another machine to command the controller.

When low-cost computers were developed, they became a natural source of input signals to the stepper motor controller. The computers provided a *pulse train* to the motor driver module. If the motor application required the stepper motor to be accelerated and decelerated, then the computer provided the properly timed pulses needed to describe the acceleration and deceleration curves. More importantly, the computer provided the calculating power required to perform complex motions. Another feature provided by computer control was the ability to synchronize the stepper motor motions with external events.

Precision motion control is a nontrivial problem and the required computer programming represents a major design effort. The problem becomes more complicated if many motors need to be controlled. The computations needed for multimotor control quickly exceed the capabilities of low-cost computers. One solution to this problem is to create dedicated hardware that can take over the detailed stepping motor control functions and be directed by a low-cost computer.

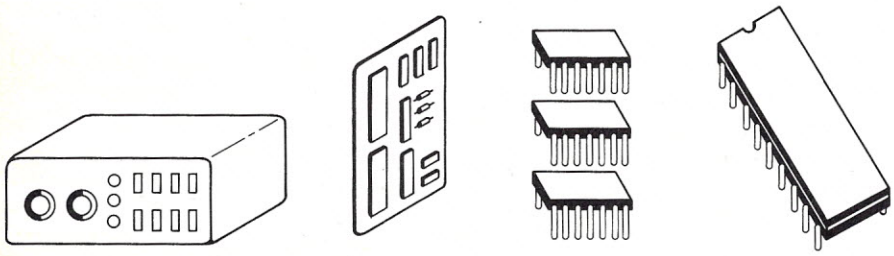


Figure 1. Motor controllers have evolved from large ungainly boxes, to boards, to single-chip controllers.

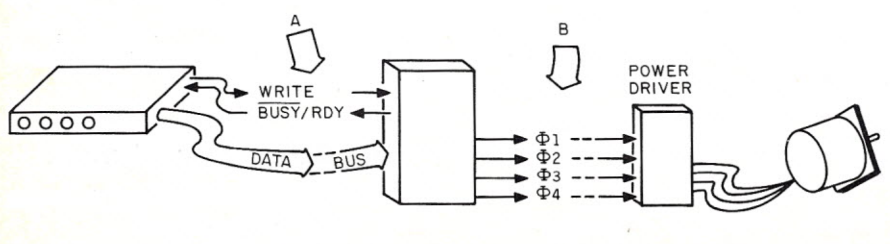


Figure 2. The CY500 (introduced in 1979) was the first single-chip controller that provided all the functions necessary for high-level stepper motor control.

THE CY500

In 1979, Cybernetic Micro Systems introduced an intelligent single-chip stepper motor controller, the CY500. The primary rationale behind the CY500 was to provide a simple interface chip and say, "Hook your computer to point A and your stepper driver to point B" as shown in Figure 2.

In addition to a simple interface, the CY500 also offers high-level functions designed for precision positioning, specifically, the ability to move a specified number of steps (relative move) and also the ability to move to a specified location (absolute move). Thus, the CY500 can be simply told to "take n steps" or "go to position n" (where $0 < n < 64K$).

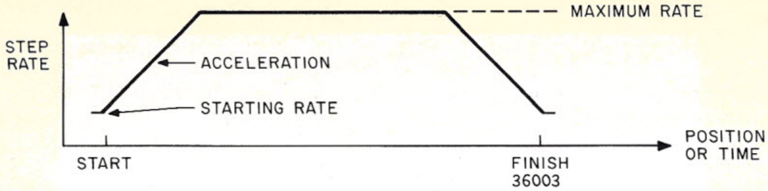


Figure 3. This graph illustrates a typical stepper motor move when a position command is given to the CY500.

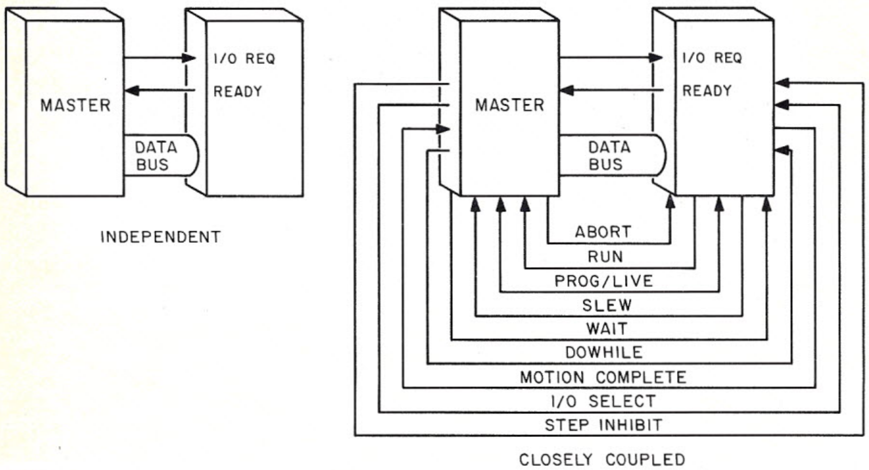


Figure 4. The CY525 can operate in two primary modes: independent and closely coupled.

The CY500 control chip accepts ASCII commands and ASCII-decimal values. This allows you to use the CY500 with high-level components (keyboards and displays) and high-level languages like BASIC. The ASCII commands consist of single alphabetic characters followed by a decimal number if applicable. For example, the command to step to position 36003 is sent to the CY500 as:

P 36003<CR>

(Where <CR> is the ASCII carriage return character, hexadecimal 0D, decimal 13.) The CY500 can also be instructed to accept binary number values. This is particularly useful when controlling the chip from machine language.

Command *sequences* can be stored on-chip. Depending on the particular CY5xx Stepper Motor Control chip, from 10 to 50 instructions can be stored. One major advantage gained from on-chip storage is the ability to define conditional branching and looping. Thus, a program sequence with decision-making capability can be stored on-chip.

Because most stepper applications operate within the framework of larger applications, the CY5xx instruction set emphasizes the ability to test *external events* as opposed to simply testing the results of internal calculations. The CY5xx devices can:

- Loop until an external event occurs.

- Wait until an external event occurs.
- Loop for a specified count of events.

In addition to sensing external signals, the CY5xx controllers also generate both specific and general-purpose control signals to be used by external subsystems. The ability to sense, signal, and synchronize, provides a very powerful subsystem *building block* for motion systems.

INTELLIGENT MOTION CONTROL

Although stepper motors can operate with considerable angular velocity or step rate, most must begin (and end) with a low step rate. Thus, a high-level controller should provide automatic acceleration and deceleration. The controller should determine when to stop acceleration and also when to begin deceleration to stop at the desired position. The CY500 has a programmable *slope* parameter that specifies the acceleration as shown in Figure 3.

As the first intelligent stepper motor controller, the CY500 was well received by customers and won a "Product of the Year" award from Electronic Products Magazine in 1980. However, as is the case with most first products, the CY500 was not perfect. Although it was designed to provide precise motor control at speeds up to 3360 steps/second, many users needed faster stepping rates.

In 1981, Cybernetic Micro Systems introduced a second-generation stepper motor controller, the CY512. The CY512 is 90 percent pin- and instruction-compatible with the CY500 and offers additional features such as: stepping rates of up to 8000 steps/second with feedback (approximately 5000 steps/second without feedback), the ability to query the position and rate registers, and a program loop command.

CY525 INTELLIGENT RAMPING MOTOR CONTROLLER

However, customers wanted still higher performance, more linear ramping (acceleration and deceleration), unlimited free-run stepping, the ability to specify complex rate profiles, etc. In response, the CY525 was created. It is 100 percent pin-compatible with the CY512 and provides additional features such as linear to optimal acceleration curves, step rates up to 10,000 steps/second, the ability to change rates in the middle of a stepping motion (of particular importance in robotic applications where external events may control the motions), and an unlimited stepping mode in

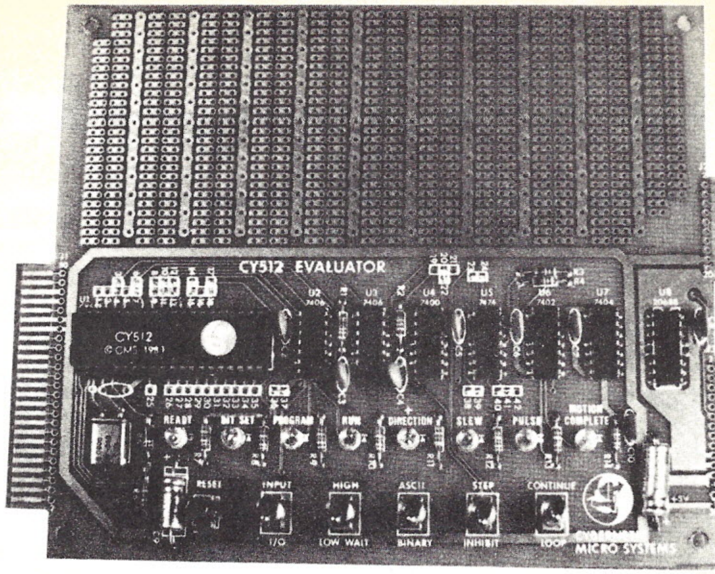


Photo 1. The CY512/Kit prototyping board available from Cybernetic Micro Systems.

TABLE 1
CY525 PIN DESCRIPTION

DESIGNATION	PIN #	FUNCTION
Prog/Live (output/input)	31	Indicates program entry mode. Commands are entered and saved, but not executed, while pin 31 is low. May be used as input to enable Live commands to be executed while a program runs.
Run (Int Req 2) (Program Complete) (output)	32	Indicates program execution mode. Commands cannot be entered while program is executing (pin 32 = low) unless the Prog line (pin 31) is used.
Motion Complete (Int Req 1) (output)	37	Signal to interrupt host at end of stepping or when position is available.
Wait (input)	38	Program Waits for this pin to go low when Until command is executed, and waits for a high signal when Wait command is executed.
Dowhile (input)	28	Is tested by T command. Program will branch to specified target if low, else it will execute next instruction.
Direction (output)	33	Indicates current stepping direction and is affected by +, −, and P commands (high = CW, low = CCW).
Pulse (output)	35	Low when step begins, high when step ends.
Slew (output)	29	Goes low when steady stepping rate has been been achieved. Will return high when ramping begins.
Abort (input)	6	Low during stepping causes the CY525 to begin ramping down to the initial step rate. If held low, the CY525 Aborts stepping at the bottom of the ramp. If the Abort line is returned high during the downramp, the CY525 ramps down and continues stepping to target position at the initial step rate.
Step Inhibit (input)	30	Inhibits stepping while held high.
Programmable Output	34	User programmable output pin.
$\Phi 1$–$\Phi 4$ (output)	21–24	Stepper drive signals.

Continued

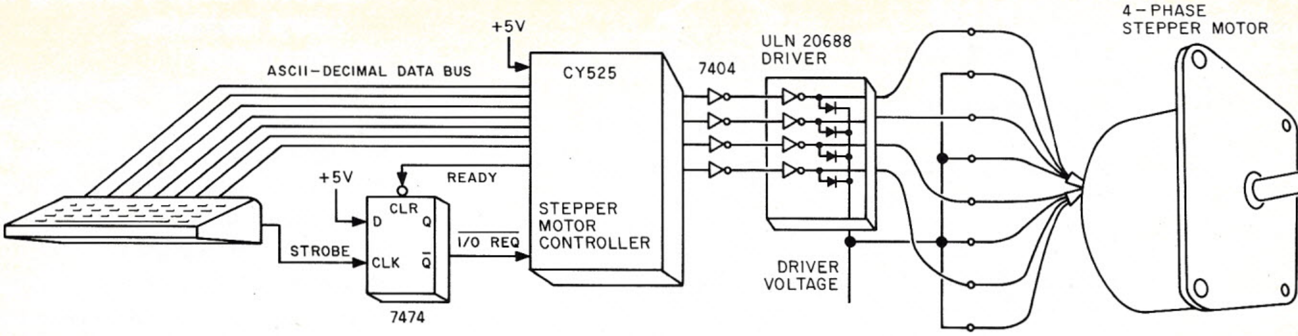


Figure 5. Simplest prototype development system for the CY525 controller chip.

which the motor steps continuously until told to stop.

The CY525 operates in one of two primary modes: independent or closely coupled. *Independent* mode is identical to the CY500 and CY512 operation. In this mode, the parameters are specified and then a motion command is issued or a program executed. The controller and master computer interaction is typically limited to signalling either that the motion is complete or the program has terminated. This high-level control moves the detail control worries from the master computer to the dedicated controller.

The second CY525 mode is called *closely coupled.* This mode provides extreme flexibility at the expense of some attention on the part of the master computer. Closely coupled mode provides the ability to change the step rate and to read the current position while the motor is stepping. This capability allows the control computer to produce complex motions.

TABLE 1

DESIGNATION	PIN #	FUNCTION
Unused	5, 9, 25	Must remain disconnected.
VCC	40	+5 volt power supply.
VDD	26	+5 volts.
VSS	7, 20	Circuit GND potential.
xtall-xtal2 (input)	2, 3	Inputs for crystal or external clock (not TTL).
Clk/15 (output)	11	This output represents the crystal frequency divided by 15. The pulse width is at least 300 nanoseconds.
$\overline{\text{Reset}}$ (input)	4	Initializes controller to power-up state.
DB0–DB7	12–19	Bidirectional parallel data bus.
$\overline{\text{I}}$/O Select	39	Indicates direction of data on the data (input) bus. Low = input to CY525. High = output from CY525, which can only be generated if CY525 has received V command. Also used to read position while stepping, i.e., on the fly.
$\overline{\text{I/O Request}}$ (input)	1	Strobe to initiate command input when writing to CY525. Initiate data output when reading from CY525. Interpretation of pin 1 is a function of I/O Select (pin 39). May be used while stepping to change the step rate on the fly.
$\overline{\text{Busy}}$/Ready (output)	27	Handshake line for command data input. Host must wait until Ready state is indicated by a high level before transferring command or data to CY525. If Run (pin 32) is low, the Prog line (pin 31) must be used to enter Live commands while a program is executing.
$\overline{\text{Instrobe}}$ (output)	8	Occurs during data input. The data on the bus must be valid until the trailing edge of Instrobe occurs.
$\overline{\text{Outstrobe}}$ (output)	10	Trailing edge indicates valid data output by CY525 on data bus.
ASCII/$\overline{\text{Binary}}$ (input)	36	Selects ASCII-decimal or binary mode of operation.

Simple Prototyping. The CY525 is an ASCII-programable peripheral controller chip designed to control stepper motors using an instruction sequence that may be stored internally in a program buffer. This feature provides the ability to program the device with an ASCII keyboard as shown in Figure 5. This greatly simplifies prototype development and experimentation. After the correct control sequence is developed, the ASCII keyboard can be replaced by a computer output port. Of course, the computer can be used initially in those systems in which keyboard programming is impractical, but most applications can usually benefit from the immediacy of the keyboard during the development phase.

The CY525 can be operated in command mode or programming mode. In command mode, the controller simply executes the command. Typing commands on the keyboard immediately causes the controller to take the appropriate action. In programming mode, a command sequence is stored in the on-chip program buffer for later execution. Since the CY525 offers stored program capability, the amount of host time and software required to perform a given task is greatly reduced.

The CY525 can function as a stand-alone device in many applications. Except for the initial program loading, the controller can stand totally separate from the host processor. In many applications, it is possible to design custom devices to load the desired program upon power-up. The

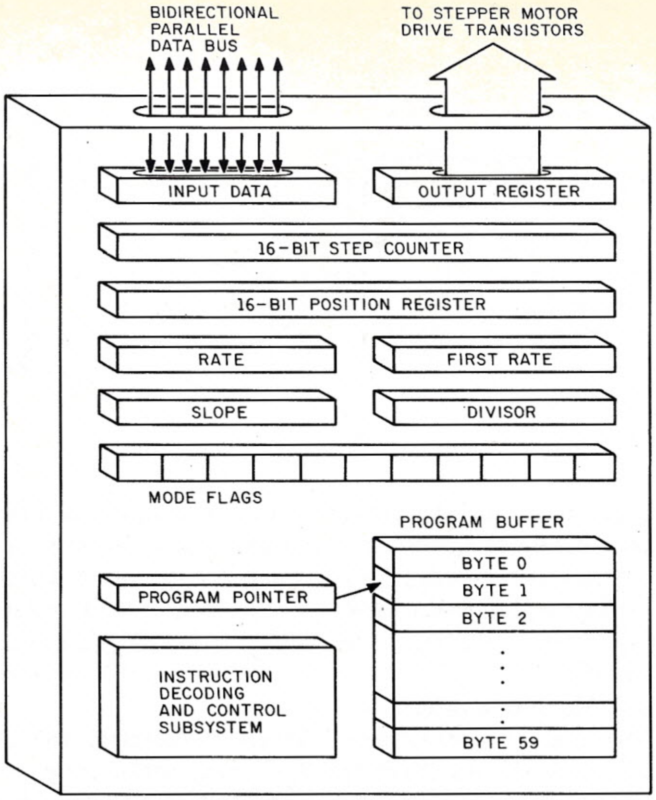


Figure 6. Schematic diagram of the CY525 Intelligent Ramping Stepper Motor Controller architecture.

TABLE 2. CY525 COMMAND SUMMARY

ASCII	NAME	INTERPRETATION
A a	Absolute	Set current location as specified
B	Bitset	Set programmable output line high
C	Clearbit	Reset programmable output line low
D d	Delay	Time delay for specified milliseconds
E	Enter	Enter program code
F f	Firstrate	Set initial step rate
G	Go	Begin relative stepping operation
H	Haltmode	Set continuous step mode of operation
I	Initialize	Turn off step drive lines, reset controller
J j	Jump	Go to specified program buffer location
L c, a	Loop	Repeat program segment for specified count
N n	Number	Set number of steps to be taken (relative)
O o	Offset	Set next stepper drive signal value
P p	Position	Set and step to target position (absolute)
Q *	Quit*	Quit entering program code, re-enter command mode. **Never followed by* <CR>
R r	Rate	Set step rate parameter
S s	Slope	Set ramp rate for slew mode operation
T t	Branch Til	Branch "Til" dowhile line goes high
U	Until	Stop execution until wait line is low
V v	Verify	Verify internal buffer contents
W	Wait	Stop executing until wait line is high
X	eXecute	Begin program execution
Z z	divisor	Divide slope by divisor parameter
+	CW	Set clockwise direction
–	CCW	Set counterclockwise direction
0	Command	Stop program execution, enter command mode
$	Label	Marker for "jump to" and "loop to" commands

programs can be triggered by external hardware. This approach removes any need for a host computer system.

CY525 ARCHITECTURE

The CY525 architecture shown in Figure 6 can be partitioned into several functional subsystems:

- Input data subsystem
- Output data subsystem
- Program parameter storage
- Mode flags and pins
- Program storage buffer, 60 bytes
- Instruction selection, decoding, and control mechanisms
- Position register

I/O Data Subsystems. The input data subsystem accepts controller commands. The output data subsystem contains the output control signals to the stepper drive circuitry and includes the associated direction and pulse timing lines.

Program Parameter Storage. The program parameter storage subsystem is used to store the step rate parameters, ramp rate parameter, and to maintain a 16-bit position register. The position register is incremented when stepping in the clockwise direction (decremented in the counterclockwise direction). The position register is used when *absolute* position commands are specified. The 16-bit step counter is used when *relative* commands are employed. The position register contents change with every step. The step counter register contents remain unchanged until a specific command is used to change them.

Mode Flags and Pins. The mode flags and mode select pins are used during command execution to perform the appropriate action or to interpret data or input signals correctly.

Program Storage Buffer. The CY525 contains a program buffer that stores an instruction sequence for execution. This provides all of the benefits of stored program execution that have made computers such powerful tools.

Instruction Decoding and Control. This subsystem performs the actual command execution.

Position Register. The CY525 contains a 16-bit position register that contains the current stepper motor location. The CY525 accepts relative and absolute position commands. However, the position register always indicates absolute position. In the CY525, this register can be read while the CY525 is stepping.

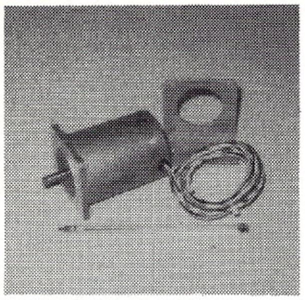
Part II of *A Third-Generation Stepper Motor Controller* discusses the CY525 architecture and provides a programming example.

Reader Feedback

To rate this article, circle the appropriate number on the Reader Service card.

75	85	95
Excellent	Good	Fair

STEPPER MOTOR SALE!



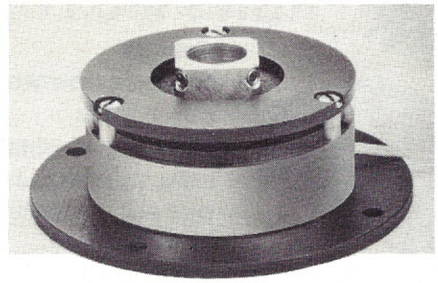
SALE PRICE $75.00

SAVE 37% OFF $119 LIST

- HIGH TORQUE: 65 OZ IN (100 OZ IN HOLD)
- SIZE: 2.2″ SQ x 3.0″ LONG (PLUS ¼″ SHAFT)
- STEP SIZE: 1.8° (200 STEPS/REV)
- LOW INDUCTANCE FOR FAST RESPONSE
- SUPPLY VOLTAGE: 1.7V TO 48V
- MOUNTING BRACKET: $9.95
- QUANTITIES ARE LIMITED

DATACRAFT
12207 AURORA NORTH
SEATTLE, WA 98133
(206) 367-9680

VISA



With hardly any noise and no asbestos — Silent Stop™ fail-safe brakes are perfect for office machines.

Disc drives, film drives, copiers, presses, printers — any office machine that might have to come to a fast, silent stop if the current fails — need our SBF series fail-safe brakes. Noise is 49 dBA max. under normal office/lab environments. No slip rings or brushes. Four models available, torque 1, 3, 7, 50 in. lb. **Call or write VERNITECH, 300 Marcus Blvd. Deer Park, NY 11729. 516/586-5100. TWX 510-277-6079.**

INTRODUCING A SINGLE-BOARD COMPUTER YOU CAN'T PASS UP

INTRODUCING THE ZINGER COMPUTER SYSTEM

Have you ever needed a small, low power easily programmable application board? This is it!

Zinger CPU • 8073 CPU with Tiny BASIC • 2K×8 RAM expandable to 12K on board • Board supports JEDEC family ROM/RAM • EEPROM programmer on board • EEPROM/RAM interchangeable • 24 line programmable parallel I/O port • Optional on-board switching power supply supplies +12 & −12V for all interface devices so that only +5V 300mA need be supplied • All devices memory mapped for ease of programming •

ASSEMBLED & TESTED $119.95

ZINGER INTERFACE BOARD

Board may be purchased with any or all combinations • 276 word speech synthesizer • 8 channel A/D converter • RTC with programmable latches & battery backup • programmable keyboard interface with 64 keys or sense switches for remote sensing • 8 char alphanumeric display or intelligent 16 char by 2 line LCD display • 24 line programmable parallel I/O port • all devices connected to interrupt register • all devices memory mapped •

UNIVERSAL AC/DC CONTROL BOARD

Controls 14 4A 200V optically isolated devices & 10 transistors for high current switching • 32 sense or drivelines • all sense/drive lines use CMOS bilateral switches so that it may be used for an 8 to 1 line 0 to 5V analog multiplexer • board designed to use one 24 line PIO port or CPU or interface board •

AND FOR HOME CONTROL...
BSR INTERFACE FOR REMOTE CONTROL OF AC DEVICES

Compatible with BSR & other wall modules sold by Radio Shack, Leviton & others • driven from a parallel port •

SCHULZ ENTERPRISES INC.
1285 LAS TUNAS DR. SAN GABRIEL, CA 91776 **(818) 287-5067**

PERSONAL ROBOTICS EXCLUSIVELY

FACTORY AUTHORIZED DEALERS

SENSATIONAL — CONTROL SYSTEM FOR HERO-1
Write, debug, edit, download programs from your Z-100, H-89, Apple II, CoCo, PC, others
MENOS System, includes RS232, 34K, OS in ROM
from Virtual Devices, Inc. List $595.00 SALE $540.00

BURN YOUR OWN RB5X PROMS
Introducing our Tiny BASIC PROM Development System
Hardware, Software, Documentation — Complete Package for 8073 CPU
VIC-20 Version .... $299.95 Commodore 64 Version $349.95
Dealer Inquiries Invited

KITS	List	SALE
MOVIT		
Screwdriver Kits—No Solder—FUN		
Avoider	44.95	40.15
Circular	67.95	63.75
Line Tracer	39.95	36.75
Medusa	27.95	26.25
Memoconcrawler	74.95	70.15
Monkey	24.95	23.65
Mr. Bootsman	30.95	29.25
Peppy	24.95	23.65
Piper Mouse	44.95	40.13
Sound Skipper	24.95	23.65
Turn Backer	39.95	36.75
HARVARD ASSOC.		
Tasman Turtle	399.00	365.00
Turtle Tot	299.00	275.00
ROBOT SHOP		
Droid Bug	129.95	122.00
Z-1	149.95	136.50
Z-2	249.95	223.50
X-1	399.95	360.00
RHINO		
Scorpion & XR-3		CALL

ROBOTS	List	SALE
ANDROBOT		
Slave to your computer		
TOPO II	1595.00	1485.00
Accessories		CALL
COMRO		
On-board 8-bit computer		
TOT		CALL
Accessories		CALL
HUBOT		
On-board 8-bit computer		
Premier 1000	3495.00	3275.00
Accessories		CALL
IOWA PRECISION		
On-board 16-bit computer		
Marvin MKI	5995.00	5785.00
PERSONAL ROBOTICS CORP.		
On-board 8-bit computer		
Ropet-HR	2499.00	2210.00
Ropet-XR	2199.00	2030.00
Accessories		CALL
RB ROBOT		
Download from your computer		
RB5X	2295.00	2058.00
Accessories		CALL

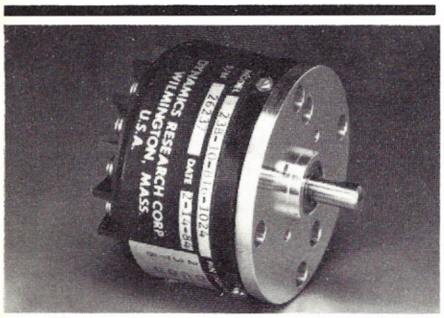
TECHNICAL INFO—WE CAN HELP. CALL

Shipping Over $200 Add 4%. $200 And Under Add 5%.
Cash With Order Deduct 3%. N.M. Orders Add 4¾% Sales Tax.
Allow 4 Weeks For Delivery.

MasterCard VISA

RIO GRANDE ROBOTICS RG
1595 W. Picacho #28, Las Cruces, N.M. 88005, Tel. (505) 524-9480

New Products

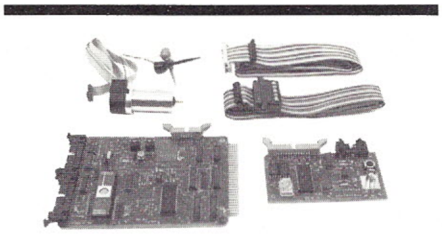


Low-Cost Industrial Encoder

The Model 23 rotary encoder is designed specifically for medium- to light-duty industrial applications. The unit features sealed bearings for maximum reliability and offers up to 4096 counts per shaft revolution. The Model 23 provides low-cost, medium resolution in a number of electrical and mechanical configurations. The unit features a single LED light source and 5, 12, and 15 VDC inputs. Output configurations include low-level sinewave, amplified sinewave, and TTL- or CMOS-compatible line drivers.

The Model 23 is OEM-priced and available for standard delivery. For more information, contact: Dave Silva, Dynamic Research Corp., 60 Concord St., Wilmington, MA 01887, telephone (617) 658-6100.

Circle 41

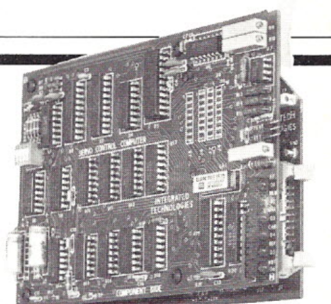


Digital Control Trainer

Galil Motion Control has announced the Digital Control Trainer (DCT 70) designed to train engineers in the field of digital servosystems. The DCT 70 is a complete motion control system which includes a motor, encoder, amplifier, and a controller. The controller closes the loop on the motion position. System stability is achieved by a digital filter and no velocity feedback is required. The digital filter parameters can be changed and the resulting effect on the system's behavior observed. Also available is a set of experiments that can be performed on the DCT 70. These are simple experiments that last about two hours each. The experiments include the effect of the digital filter on the step response, the frequency response of digital control systems, and others.

For further information, contact: Dr. Jacob Tal, Galil Motion Control, 1916-C Old Middlefield Way, Mountain View, CA 94043, telephone (415) 964-6494.

Circle 42

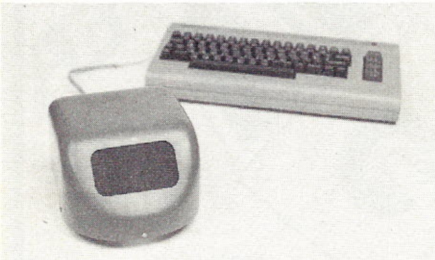


Servocontrol Computer

The Integrated Technologies Model 102A Intelligent Servocontrol Computer provides the capability to control a DC servoposition or velocity loop from the STD Bus. The 102A uses an incremental encoder for position feedback and the STD Bus for commands. By using sophisticated control algorithms, the 102A develops the appropriate analog velocity signal to feed a standard servo amplifier. A complete memory check is made after power-up, and feedback inputs are verified constantly. Any faults are indicated by a bank of six LEDs. An isolated fail-safe watchdog timer signals external equipment of emergency conditions.

The price of the 102A in quantities of ten is $433 each. For information, contact: Brian Quinn, Integrated Technologies, Inc., 444 West Maple, Building F, Troy, MI 48084, telephone (313) 362-4466.

Circle 43



Mobile Nomad

The Nomad robot from Genesis Computer Corp. is an affordable robot for use with the Commodore 64 microcomputer. Nomad is a versatile educational tool for teaching programming logic to young children or computer novices. While teaching Nomad how to move about, the student learns the same kind of logic necessary for writing computer programs. Nomad is also an excellent entertainment companion that is sure to keep experimentalists and recreational programmers busy for hours on end.

Nomad is constructed of a sturdy aluminum chassis with a durable thermo-formed plastic body that resists dents and scratches. It is driven by precision stepper motors and can move forward, backward, right, and left. Nomad has ultrasonic eyesight, which allows it to detect objects in its path or to do ranging. Nomad's special Robot Control Language allows the user to easily create complex patterns of movement. Nomad plugs into the User Port (RS232) and has a 25-foot cord, which allows for movement within a typical two-room area or classroom. It comes with its own power supply. An optional BASIC enhancement cartridge adds Nomad control commands such as AHEAD, BACK, LEFT, and RIGHT directly to the Commodore 64's BASIC.

Nomad is available immediately for the Commodore 64 and has a suggested retail price of $179.95. The optional extended BASIC cartridge retails for $39.95. A Radio Shack Color Computer model is available exclusively through Frank Hogg Laboratories of Syracuse, New York and an Apple IIe model will also be available by late December.

For more information, contact: Genesis Computer Corp., Ben Franklin Technology Center, Lehigh University, Bethlehem, PA 18015, telephone (215) 861-0850.

Circle 44

iLISP For CP/M Computers

Computing Insights has announced iLISP for for Z80-based microcomputers. iLISP offers all the traditional advantages of LISP programming including extensibility, modularity, and a flexible and highly interactive programming environment.

iLISP runs under the CP/M 2.2 operating system. It is based on the Lisp dialect called SCHEME and offers advanced Lisp features including runtime Lisp macros and input-time READ macros, stack free execution of *tail recursive* functions, complete access to the CP/M file directory, sequential and byte addressable random I/O to disk files, both floating point and integer arithmetic including sine, cosine, arctangent, and random numbers, programmer control of the executive, error handling, and startup functions, a simple assembly-language interface, trace and break debugging utilities.

The iLISP utilities are fully documented and include all source code. Utilities include a Lisp list editor, a function library, a pretty-print utility, and the classic Eliza program.

iLISP is available on both 8 in. and popular 5½ in. disk formats (including Kaypro, Morrow, Zenith, and Osborne). The list price is $49.95. For more information, contact: Computing Insights, PO Box 4033, Madison, WI 53711.

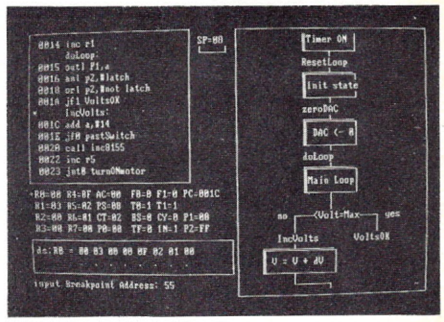
Circle 45

RB5X Development System

Rio Grande Robotics has released a PROM Development System for the RB5X Robot. The system includes a Tiny BASIC editor for writing programs to run on the RB5X's NSC 8073 processor, a terminal program to download the program to the robot or to a PROM burner, PROM burning hardware and software, and extensive documentation. The system can be used to program 2716 and 2732 EPROMs. It runs on the VIC-20 and Commodore 64 microcomputers. The system will be available from the company or through robotics retailers for $299.

For more information, contact: Rio Grande Robotics, 1595 W. Picacho #28, Las Cruces, NM 88005, telephone (505) 524-9480.

Circle 46



8048 Simulator for IBM PC

The Sim-8048 Simulator/Debugger from Cybernetic Micro Systems allows execution and debugging of machine code for the 8048 family of microcomputers on the IBM PC (128 Kbytes of programmable memory required). The dynamic display and interactive nature of the program should cut 8048 program development by an order of magnitude.

The Sim-8048 dynamically displays the source code, register values, flags, I/O pins, and program branches in windows. In addition to the usual updating of register contents, the unique flowgraph scrolls through the window as the program executes, allowing the user to visually follow the code. Sim-8048 supports the timer-counter and external interrupts and even the use of 8155 external memory.

The 8048 code is derived from a HEX file produced by the CYS-8049 cross-assembler or equivalent. The Sim-8048 provides 50 commands consisting of single alphabetic keys or control characters that provide register value declaration, trap specification, breakpoint specifications, single step, auto-step, fast execution, etc. The simulator provides access to all of the 8048 memory, including data, code, and external space. Selected data is displayed and can be easily altered. The Sim-8048 displays both the source (symbolic) names and the numeric values. Online help is always available.

The Sim-8048 is priced at $395. A demo disk and manual are available for $39.50. For more information, contact: Cybernetic Micro Systems, PO Box 3000, San Gregorio, CA 94074, telephone (415) 726-3000.

Circle 47

Voice-Controlled Applications

Arctec Systems has introduced voice-controlled video games for personal computers for use with the Micro-Ear product line. According to the company, they are the first voice-controlled video games available for personal computers. Micro-Ear is a speaker- and language-independent voice-recognition and command system that can be used with all personal computers capable of RS232 communications.

Up to 256 words are selected and defined by the user and trained by the user's voice. Speaking a single word can activate an operation, cause a character to appear on the screen, execute a program, or execute a series of instructions. The user's voice patterns and the communications software are retained in the battery-backed Micro-Ear memory.

Micro-Ear plugs into an RS232 interface port and uses the standard RTS/CTS protocol. The system retails for $579 and comes with a microphone, 9V power supply, user's guide and demonstration programs (including video game programs) for IBM PC and Apple II series computers. Software for the IBM PC also includes EAR DOS, which allows voice control from inside virtually any applications software.

For more information, contact: Arctec Systems, 9104 Red Branch Rd., Columbia, MD 21045, telephone (301) 730-1237.

Circle 48

Stepping Motor Controller

The EM-8944 provides a complete interface to a pair of open-loop stepping motor drives. The EM-8944 provides two fully independent axes per card. Onboard acceleration/deceleration and position counter logic handle all real-time axis control tasks of acceleration/deceleration and position counting. The logic implements a velocity-controlled servo rather than a preset indexer, allowing both contouring and positioning axes control.

The Multibus processor sets direction, loads 8-bit velocity commands, monitors a status register, and services position counter overflows. The EM-8944 accelerates or decelerates to each new velocity, signals "AT SPEED" with status register bits, maintains the 8 least significant bits of position count, and signals counter over/under flows with status flags and optional interrupt requests. Inverse exponential deceleration provides optimum motor performance.

Each axis has individually adjustable maximum slew speed and acceleration rates. The acceleration profile is a digitally generated rising exponential between zero and maximum slew speed, such that final acceleration is 1/15 of initial acceleration. Deceleration is the mirror image of acceleration, matching the braking torque demand to the increasing motor torque available as speed decreases. Thus, both acceleration and deceleration motor torques can be more closely matched to the torque/speed motor characteristics of typical stepping motors.

The EM-8944 interface is priced at $785. For complete technical details, contact: Symbicon Associates, Inc., 89 Route 101A, Amherst, NH 03031, telephone (603) 673-8898. Circle 49

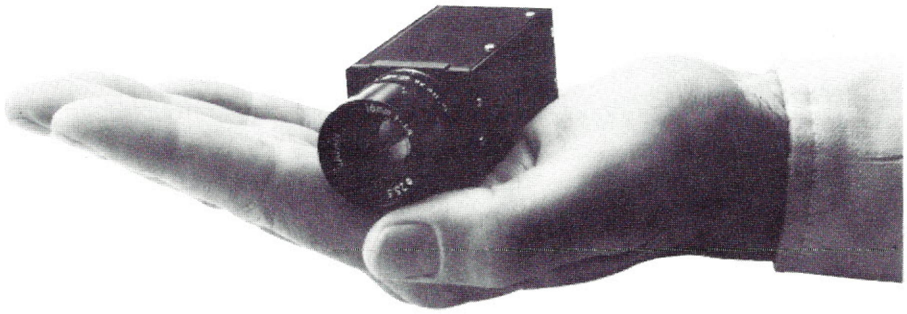
MasterFORTH

MasterFORTH is the latest generation of MicroMotion Forth interpreters and meets all provisions of the new Forth-83 International Standard. MasterFORTH is a complete professional programming language and includes built-in macroassembler with local labels, a screen editor, and a string handling package. The input and output streams are redirectable and the system uses the host operating file structure. A floating-point option is also available. The package includes *Forth Tools*, a 200-page textbook, a technical reference manual, and a complete listing of the MasterFORTH nucleus.

MasterFORTH is available for Z-80 CP/M 2.x operating systems and Apple II/II+/IIe systems. It retails for $100 or $140 with the floating point option. For more information, contact MicroMotion, 12077 Wilshire Blvd. #506, Los Angeles, CA 90025, telephone (213) 821-4340.

Circle 50

Ultraminiature CCD Video Camera



A line of ultraminiature CCD (charge coupled device) video cameras has been introduced by the Component Products Division of Sony Corp. The XC-37, XC-38, and XC-47 CCD cameras eliminate problems of lag, image burning, geometric distortion, and microphonic noise associated with pick-up tube cameras. The XC series cameras are highly sensitive high-resolution cameras based on Sony's patented CCD solid-state imaging technology.

The cameras perform well in poorly lit areas, in magnetic fields, and in areas subject to vibration and shock. These performance strengths, combined with the cameras' small size (one-third that of pick-up tube cameras) make Sony's CCD video cameras ideal for industrial automation processes such as recognizing patterns and positioning robot arms. The cameras' small power consumption (2.3 W) and small size facilitate multiple installation on machine vision systems that must quickly inspect complex surface structures.

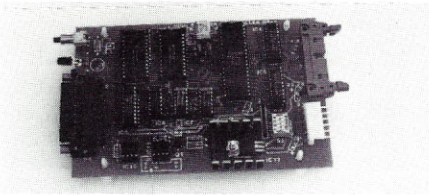
For information, write: Sony Component Products Division, 15 Essex Road, Paramus, NJ 07652. Or call Joe Thorsen or Alan Penchansky at (212) 575-1976.

Circle 51

Circle 6

New Products

Improved High Stepper



Cyberpak Co. has extended the capabilities of its High Stepper line with the HS-1, a BASIC programmable microcomputer that is plug-compatible with its HS-2 Dual Stepper Motor Controller. The HS-1 is based on Zilog's Z8 microcomputer chip.

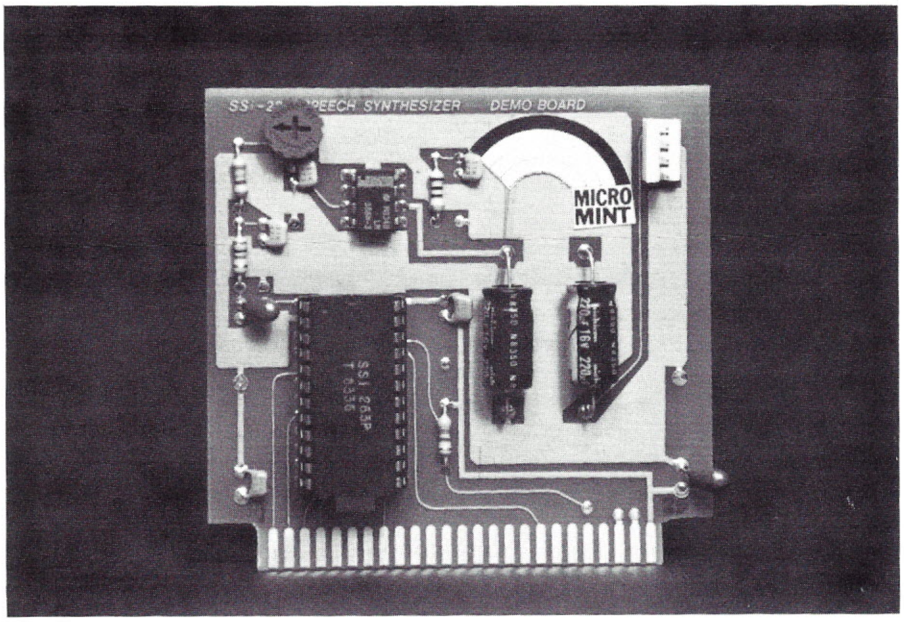
The HS-1 offers an opto-isolated RS232 serial port with a hardware-selectable, software-readable data transfer rate. This allows the HS-1 to be used in a variety of configurations. The HS-1 can be used with a personal computer, and allows the saving of BASIC programs over its serial port.

Two 28-pin memory sockets allow a variety of memory configurations. The maximum memory is 30 Kbytes. The standard HS-1 comes with 2 Kbytes of read-only memory and 2 Kbytes of programmable memory. The HS-1's standard read-only memory contains software for driving the HS-2 motor controller. With it, you can perform stepping motions through function calls from the BASIC interpreter.

Software functions allow you to specify which of the two motors to move, A or B; the direction; full-step or half-step mode; the number of steps to move; and the step rate in steps per second. This embedded software is written in Z8 assembly language to achieve fast and efficient motion control, yet allow full use of the BASIC interpreter.

The HS-1, priced at $169, is part of a series of products that combine to form a complete motion control system. Motor controllers, power supplies, stepper motors, and cables are also available. For further information, contact: Cyberpak Co., PO Box 38, Brookfield, IL 60513, telephone (312) 387-0802.

Circle 52



Sweet Talker II

Sweet Talker II is the third generation speech synthesizer from Micromint. The Sweet Talker II is based on the newest development in phonetic speech synthesis—the SSI 263 integrated circuit from Silicon Systems, Inc.

The SSI 262 chip can be configured for various levels of intelligibility. At amazingly low data rates of no more than 400 bps, the Sweet Talker II can synthesize human singing complete with vibrato. Intonation, inflection, and filtration are all under program control and phoneme values can be set dynamically on five internal registers (a total of only 40 bits). Sweet Talker II is designed to work with the Apple II+ or IIe but it can also be used on many other computers. Appropriate controls are provided for address mapping with several buses. Its features include 256 phoneme equivalents, 4096 pitch variations, and an onboard 1 W amplifier with volume control. There are five 8-bit internal registers, four modes of handshaking, and eight articulation rates. Finally, the Sweet Talker II also has 16 speed settings and amplitude levels as well as 255 settings of the vocal-tract filter and frequency response.

A custom-designed text-to-speech algorithm employs a rule-based method of pronunciation that encompasses whole-word, morpheme, and letter rules in character-specific subtables. A utility routine is provided for changing, editing, or redefining the rules. The utility can adjust the size of the main routine dependent on whether rules or characters in rules have been inserted or deleted. This makes the availability of foreign languages and dialects as simple as redesigning the rule table.

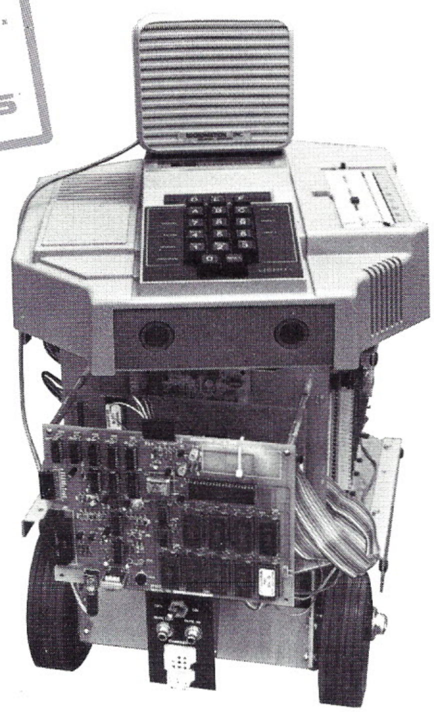
For more information about the Sweet Talker II, contact Micromint, Inc., 561 Willow Ave., Cedarhurst, NY 11516, telephone (516) 374-6793.

Circle 53

Classified Advertising

boost your HERO-1 to powerful new levels

MICROMATION, INC. is now arctec systems



Arctec Systems' whole new line of power-packed peripherals helps your Heathkit HERO-1™ personal robot reach its greatest potential. And with our dynamic software, you can even *talk* to your robot.

Meet Our Peripherals and Software:

HERO MEMCOM™ Board

Develop great, new programs for your HERO-1 robot using your personal computer — and at the same time *boost the robot's memory almost ten-fold with an additional 30K of RAM!*

Our MEMCOM Board comes with an RS-232 serial port for direct communications between your robot and practically *any* computer . . . two 8-bit bi-directional parallel ports for running peripherals — plus two 16-bit timers. Now — make superfast robot-computer data transfers!

You get fast and easy-to-use communications software, for uploading and downloading programs via the serial port . . . you can save programs when the robot is in its "sleep" mode . . . save programs on disk for later use — and all the hardware and cables needed for easy installation are included. Price: $345.

Talk to Your Robot—with Our Voice Command System

Connect the versatile **Voice Command System** to the HERO MEMCOM Board, and program your robot's movements *by voice* alone. You get **VOREC** (Voice Recognition Board): a precise, speaker-dependent recognizer, with a flexible vocabulary of up to *256 words* you train to your voice patterns. Plus **VOCOL:** the high-level machine language that gives your robot immediate *and* deferred execution modes. Price: $595. Voice Command System with HERO MEMCOM Board: $920.

Apple-HERO Communicator

Comes with all the hardware and software needed to hook up an Apple™ computer to your robot and HERO MEMCOM Board. Price: $159.

And More

Storyteller . . . Poet . . . Robot Language Macro . . . 6808 Cross Disassembler . . . Text-to-Phoneme Conversion . . . RS-232 Interface . . . Downloader . . . Development Disk for CP/M . . . Robot Cross-Assembler Package . . . Prom Adapter Extender Board . . . Polaroid Rangefinder. Arctec Systems has a full range of exciting add-ons for your HERO-1 personal robot.

Act Right Now

For more detailed information on the incredible range of Arctec add-ons *available today*—write or call.

arctec systems™

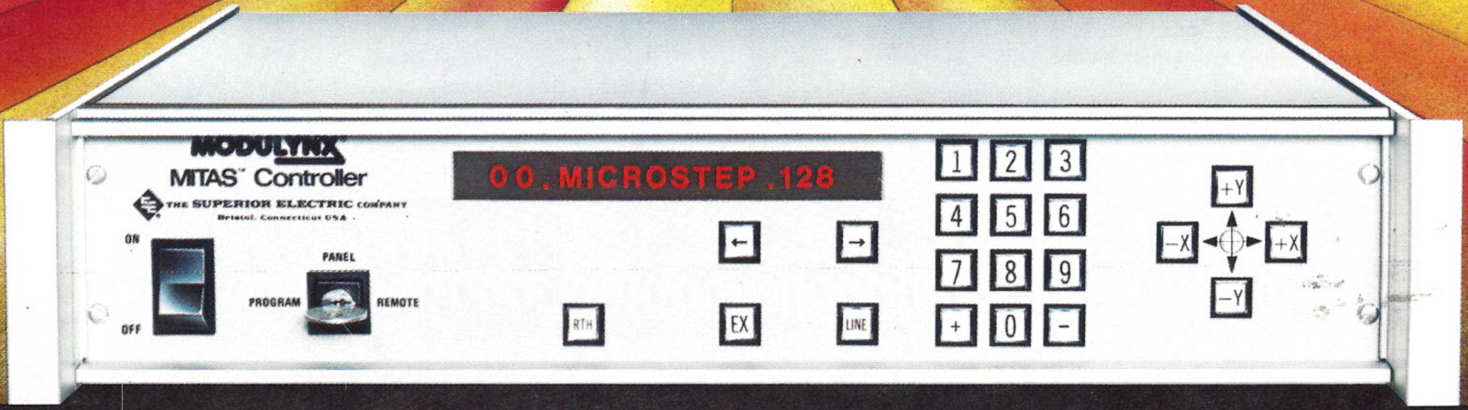
9104 Red Branch Road Columbia, Maryland 21045
(301) 730-1237 • Telex 87-781

Circle 4

THE GOLDEN TOUCH TO AUTOMATION
THE SIMPLIFIED MOTION CONTROL SYSTEM

FEATURING . . . USER FRIENDLY-MENU DRIVEN PROGRAMMABILITY • TEACH MODE • MICROSTEPPING RESOLUTION TO .014°—25600 STEPS/REV • 16 BIT MICROPROCESSOR CONTROL • NONVOLATILE MEMORY • 99 PROGRAMMABLE MOVES • WIDE RANGE OF DRIVES • AUTO/MANUAL OPERATION • MEMORY EDITING • RELAY OUTPUTS AND EXTERNAL I/O CONTROL • SLO-SYN STEPPING MOTORS • TORQUE FROM 25 OZ.-IN. TO 2.5 H.P. • PROGRAM SECURITY • SELF DIAGNOSTICS • TABLE TOP OR 19' RACK MOUNTABLE • INCREMENTAL OR ABSOLUTE OPERATION • BACKLASH COMPENSATION • JOYSTICK OPERATIONS • USER FRIENDLY-MENU DRIVEN PROGRAMMABILITY • TEACH MODE • MICROSTEPPING RESOLUTION TO .014°—25600 STEPS/REV • 16 BIT MICROPROCESSOR CONTROL • NONVOLATILE MEMORY • 99 PROGRAMMABLE MOVES • WIDE RANGE OF DRIVES • AUTO/MANUAL OPERATION • MEMORY EDITING • RELAY OUTPUTS AND EXTERNAL I/O CONTROL • SLO-SYN STEPPING MOTORS • TORQUE FROM 25 OZ.-IN. TO 2.5 H.P. • PROGRAM SECURITY •

MITAS™



MODULYNX® Intelligent Two-Axis System
for stepping motor control

Superior Electric
The leader in motion control technology

For more information or the name of your nearest MODULYNX/MITAS Distributor, call (203) 582-9561, ext. 383 or write to The Superior Electric Company, 383 Middle Street, Bristol, CT, 06010

Circle 31